Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH

Elektronische Gesundheitskarte und Telematikinfrastruktur

Spezifikation des Card Operating System (COS)

Elektrische Schnittstelle

Version:	3.13.0
Revision:	166392
Stand:	02.10.2019
Status:	freigegeben
Klassifizierung:	öffentlich
Referenzierung:	gemSpec_COS

Dokumentinformationen

Es handelt sich hier um eine veränderte Fassung des Dokumentes für Karten der Generation 2.2.

Dieses Kapitel erläutert den Farbcode der farblichen Hinterlegung von Änderungen beim Vergleich von [gemSpec_COS] in der Version 3.12.0 vom 15.05.2019 gegenüber der aktuellen Version.

1) blaue Hinterlegung: Editorielle Änderungen, die sich weder auf Implementierungen durch Hersteller, noch auf Testimplementierungen auswirken.

2) gelbe Hinterlegung: Änderungen, die sich nicht auf die Implementierung durch Hersteller, wohl aber auf gematik-Artefakte (Spezifikationen, Testimplementierung, …) auswirken.

3) orange Hinterlegung: Änderungen, die sich auf die Implementierung durch Hersteller und/oder die gematik-Artefakte auswirken.

Es ergibt sich eine Kritikalität von niedrig=ohne über blau, gelb nach orange. Im Zweifelsfall wird eine höhere Kritikalität, also eine stärkere Warnfarbe gewählt.

Die wichtigsten Änderungen gegenüber der Betriebssystemspezifikation der Generation 2.1 sind:

1. Das Kommando GET RANDOM ist nicht länger an die Option_logische_Kanäle gebunden, sondern verpflichtend.
2. Die Algorithmen rsaDecipherPKCS1_V1_5 und rsaEncipherPKCS1_V1_5 wurden aus dem normativen Funktionsumfang entfernt. Für Ver- und Entschlüsselung mittels RSA-Kryptographie steht nach wie vor rsaEncipherOaep und rsaDecipherOaep zur Verfügung und die Ver- und Entschlüsselung mittels elliptischer Kurven ist von dieser Änderung nicht betroffen.
3. Die Priorisierung der Trailer wurde für die Kommandos in 14.6 so geändert, dass die Blockade eines Passwortobjekts auch bei bestehendem Transportschutz zuverlässig erkannt wird.
4. Die Funktionspakete zu Option_DES und Option_RSA_CV wurden mitsamt den davon abhängigen Anforderungen entfernt.
5. Die Performanzvorgaben wurden aktualisiert.

Die wichtigsten Änderungen gegenüber der Betriebssystemspezifikation der Generation 2 sind:

6. Der DES-Algorithmus wurde aus dem normativen Umfang in ein optionales Funktionspaket verschoben, siehe (N000.030), entsprechende Performanzvorgaben wurden entfernt.
7. CV-Zertifikate basierend auf dem RSA-Algorithmus wurden in ein optionales Funktionspaket verschoben, siehe (N000.032) entsprechende Performanzvorgaben wurden entfernt.
8. Performanzvorgaben angepasst, das bedeutet:

   a. Dort, wo die zugelassenen G2 COS deutlich performanter sind, wurden strengere Werte gesetzt und
   b. dort, wo die Performanzvorgaben viel zu streng waren, wurden sie an den Stand der Technik angepasst.
9. Anforderungen zur Kürzung von *keyReferenceList* im Kommando LIST PUBLIC KEY präzisiert.
10. Die RSA-Schlüsselgenerierung wurde aus dem normativen Umfang in ein optionales Funktionspaket verschoben, siehe (N000.034).

Die wichtigsten Änderungen gegenüber den Betriebssystemspezifikationen der Generation 1 sind:

11. Alle Betriebssystemspezifikationen der Generation 1 wurden in diesem Dokument zusammengefasst.
12. Alle SRQs zu den Betriebssystemspezifikationen wurden integriert.
13. Die kontaktlose Datenübertragung und die Datenübertragung gemäß [ISO/IEC 7816-12] wurden aufgenommen.
14. Die Kryptographie gemäß [FIPS 197] (AES) und [BSI-TR-03111] (elliptische Kurven) wurden dem normativen Teil hinzugefügt.
15. Für RSA-Schlüssel ist die Moduluslänge von 3072 bit zu unterstützen.

Dokumentenhistorie

Version	Stand	Kap./ Seite	Grund der Änderung, besondere Hinweise	Bearbeitung
2.2.0	R 0.5.2 R 0.5.3		Die Dokumentenhistorie zu freigegebenen Dokumenten im Release 0.5.2 bzw. 0.5.3 und Vorgängerversionen ist in den dort gültigen Dokumenten ausführlich dargestellt.	gematik
3.0.0	20.09.12		Zusammenfügen aller Generation 1 Betriebssystemspezifikationen Einarbeitung aller SRQs zu Betriebssystemspezifikationen der Generation 1 Einarbeitung der Generation 2 Anforderungen, unter anderem: Kontaktlose Datenübertragung, Datenübertragung gemäß [ISO/IEC 7816-12], AES, elliptische Kurven, RSA Moduluslänge 3072 bit	gematik
3.1.0	30.01.13		Anpassung des normativen Umfangs, Fehlerkorrekturen	gematik
3.2.0	22.10.13		Fehlerkorrekturen, Präzisierung PACE, Einarbeitung von Kommentaren	PL P71

3.2.1	11.12.13		LIST PUBLIC KEY Kommando, Cachen von importierten Schlüsseln, Einarbeitung Kommentare	gematik
3.4.0	21.02.14		Fehlerkorrekturen	gematik
3.5.0	01.04.14		Fehlerbereinigung	gematik
3.6.0	06.06.14		Öffentliche RSA 3072 aus normativem Umfang entfernt, Fehlerbereinigungen	gematik
3.7.0	26.08.14		Einarbeitung Iteration 4	gematik
3.8.0	17.07.15		Folgende Errata eingearbeitet: R1.4.2	Technik / SPE
3.9.0	24.08.16		Anpassungen zum Online-Produktivbetrieb (Stufe 1)	gematik
3.10.0	21.04.17		DES aus dem normativen Umfang entfernt	gematik
3.11.0	14.05.18		RSA Schlüsselgenerierung aus normativem Umfang entfernt	gematik
3.12.0	15.05.19		Einarbeitung C_6725, GET RANDOM; rsaDecipherPKCS1_V1_5 und rsaEncipherPKCS1_V1_5 entfernt, PasswordBlocked höher priorisiert	gematik
			Einarbeitung P20.1	gematik
3.13.0	02.10.19		freigegeben	gematik

Inhaltsverzeichnis

1 Einordnung des Dokuments (informativ) 21
1.1 Zielsetzung 21
1.2 Zielgruppe 22
1.3 Geltungsbereich 22
1.4 Abgrenzung des Dokuments 23
1.5 Methodik 24
1.5.1 Nomenklatur der Präfixe 24
1.5.2 Nomenklatur Verschiedenes 24
1.5.3 Normative und informative Abschnitte 26
1.5.4 Komponentenspezifische Anforderungen 26
1.5.5 Verwendung von Schlüsselworten 27
2 Optionen (normativ) 29
3 Systemüberblick (informativ) 32
4 Lebenszyklus von Karte und Applikation (informativ) 33
5 Datentypen und Datenkonvertierung (normativ) 34
5.1 BitLength Anzahl Bit in einem Bitstring 34
5.2 OctetLength Anzahl Oktett in einem Oktettstring 35
5.3 I2OS Integer nach Oktettstring 35
5.4 OS2I Oktettstring nach Integer 36
5.5 OS2P Oktettstring nach Punkt (uncompressed encoding) 37
5.6 P2OS Endlicher Punkt nach Oktettstring 37
5.7 Extrahiere führende Elemente 38
5.7.1 Extrahiere führende Bits 38
5.7.2 Extrahiere führende Oktette 38
5.8 PaddingIso 39
5.9 TruncateIso 39
5.10 MGF Mask Generation Function 40
5.11 RAND Zufälliger Oktettstring 41
5.12 ceiling, Aufrunden einer reellen Zahl 42
5.13 floor, Abrunden einer reellen Zahl 42
6 Kryptographische Algorithmen (normativ) 43

6.1 Hash-Algorithmen 43
6.1.1 SHA-1 43
6.1.2 SHA-256 43
6.1.3 SHA-384 44
6.1.4 SHA-512 44

6.2 Schlüsselvereinbarung 45
6.2.1 Verhalten für 3TDES-Schlüssel, Option_DES 45
6.2.2 Vereinbarung von AES-128-Schlüsseln 46
6.2.3 Vereinbarung von AES-192-Schlüsseln 46
6.2.4 Vereinbarung von AES-256-Schlüsseln 47
6.2.5 Schlüsselableitung aus einer Card Access Number 48

6.3 Symmetrischer Basisalgorithmus für Vertraulichkeit 48
6.3.1 Symmetrische Verschlüsselung eines Datenblocks, Option_DES 49
6.3.1.1 Verschlüsselung mittels DES, Option_DES 49
6.3.1.2 Verschlüsselung mittels 3TDES, Option_DES 49
6.3.2 Symmetrische Entschlüsselung eines Datenblocks, Option_DES 50
6.3.2.1 Entschlüsselung mittels DES, Option_DES 50
6.3.2.2 Entschlüsselung mittels 3TDES, Option_DES 50
6.3.3 Symmetrische Verschlüsselung eines Datenblocks, AES 51
6.3.4 Symmetrische Entschlüsselung eines Datenblocks, AES 51

6.4 Asymmetrischer Basisalgorithmus RSA 52

6.5 Asymmetrischer Basisalgorithmus elliptische Kurven 53

6.6 Datenauthentisierung 53
6.6.1 MAC-Generierung 53
6.6.1.1 Generierung Retail-MAC, Option_DES 53
6.6.1.2 Generierung CMAC ohne internes Padding vor der CMAC-Berechnung 54
6.6.1.3 Generierung CMAC mit internem ISO-Padding vor der CMAC-Berechnung 55
6.6.2 MAC-Prüfung 55
6.6.2.1 Prüfung Retail-MAC, Option_DES 55
6.6.2.2 Prüfung CMAC ohne internes Padding vor der CMAC-Berechnung 56
6.6.2.3 Prüfung CMAC mit internem Padding vor der CMAC Berechnung 56
6.6.3 Signaturberechnung 57
6.6.3.1 Signaturberechnung mittels RSA 57
6.6.3.1.1 RSA, ISO9796–2, DS1, SIGN, Option_RSA_CVC 57
6.6.3.1.2 RSA, SSA, PKCS1–V1_5 58
6.6.3.1.3 RSA, SSA, PSS 59
6.6.3.1.4 RSA, ISO9796–2, DS2, SIGN 59
6.6.3.1.5 RSASSA–PSS–SIGN 60
6.6.3.2 Signaturberechnung mittels ELC 60
6.6.4 Signaturprüfung 61
6.6.4.1 RSA, ISO9796–2, DS1, VERIFY, Option_RSA_CVC 61
6.6.4.2 Signaturprüfung mittels elliptischer Kurven 63

6.7 Vertraulichkeit von Daten, symmetrischer Fall 64
6.7.1 Symmetrische Verschlüsselung 64
6.7.1.1 Verschlüsselung 3TDES, Option_DES 64

6.7.1.2 Verschlüsselung AES *65*
6.7.2 Symmetrische Entschlüsselung 65
6.7.2.1 Entschlüsselung 3TDES, Option_DES *65*
6.7.2.2 Entschlüsselung AES *66*
6.8 Vertraulichkeit von Daten, asymmetrischer Fall 67
6.8.1 Asymmetrische Verschlüsselung 67
6.8.1.1 RSA, ES, PKCS1 V1.5 ..... *67*
6.8.1.2 RSA, OAEP, Verschlüsselung *67*
6.8.1.3 Elliptic Curve Key Agreement *68*
6.8.1.3.1 Elliptic Curve Key Agreement Point Sab 68
6.8.1.3.2 Elliptic Curve Key Agreement Value Zab 69
6.8.1.4 ELC Verschlüsselung *69*
6.8.2 Asymmetrische Entschlüsselung 70
6.8.2.1 RSA, ES, PKCS1 V1.5, Decrypt *70*
6.8.2.2 RSA, OAEP, Decrypt *70*
6.8.2.3 ELC Entschlüsselung *71*
7 CV-Zertifikat (informativ) 73
7.1 CV-Zertifikat für RSA-Schlüssel 73
7.1.1 Bestandteile eines CV-Zertifikats für RSA-Schlüssel 73
7.1.1.1 Certificate Profile Identifier (CPI) *73*
7.1.1.2 Certification Authority Reference (CAR) *73*
7.1.1.3 Certificate Holder Reference (CHR) *74*
7.1.1.4 Certificate Holder Autorisation (CHA) *74*
7.1.1.5 Object Identifier (OID) *74*
7.1.1.6 Öffentlicher Schlüssel *74*
7.1.1.6.1 Modulus 74
7.1.1.6.2 Öffentlicher Exponent 74
7.1.2 Zertifikatsprofile für RSA-Schlüssel 74
7.1.2.1 CV-Zertifikat für CA-Schlüssel *74*
7.1.2.2 CV-Zertifikat für Authentisierungsschlüssel *75*
7.1.3 Struktur und Inhalt eines CV-Zertifikats für RSA-Schlüssel 76
7.2 CV-Zertifikate für ELC-Schlüssel 77
8 Objekte 78
8.1 Diverse Attribute (normativ) 78
8.1.1 File Identifier 78
8.1.2 Short File Identifier 78
8.1.3 Life Cycle Status 78
8.1.4 Zugriffsregelliste 79
8.1.5 Rekord 80
8.1.6 SE-Identifier 81
8.1.7 PIN 81
8.1.8 Datum 82
8.2 Schlüsselmaterial (normativ) 82
8.2.1 Symmetrische Schlüssel 82
8.2.1.1 3TDES-Schlüssel, Option_DES *82*

8.2.1.2 AES-128-Schlüssel *83*
8.2.1.3 AES-192-Schlüssel *83*
8.2.1.4 AES-256-Schlüssel *83*
8.2.2 Domainparameter für elliptische Kurven 83
8.2.3 Privater Schlüssel 85
8.2.3.1 Privater RSA-Schlüssel *85*
8.2.3.2 Privater ELC-Schlüssel *85*
8.2.4 Öffentlicher Schlüssel 85
8.2.4.1 Öffentlicher RSA-Schlüssel *86*
8.2.4.2 Öffentlicher ELC-Schlüssel *86*
8.2.5 Transportschutz für ein Passwort 86
8.3 File (normativ) 87
8.3.1 Ordner 87
8.3.1.1 Applikation *89*
8.3.1.2 Dedicated File *90*
8.3.1.3 Application Dedicated File *90*
8.3.2 Datei 90
8.3.2.1 Transparentes Elementary File *92*
8.3.2.2 Strukturiertes Elementary File *93*
8.3.2.2.1 Linear variables Elementary File 95
8.3.2.2.2 Linear fixes Elementary File 95
8.3.2.2.3 Zyklisches Elementary File 96
8.3.3 File Control Parameter 96
8.4 Reguläres-Passwort (normativ) 99
8.5 Multireferenz-Passwort (normativ) 103
8.6 Schlüsselobjekt (normativ) 104
8.6.1 Symmetrisches Authentisierungsobjekt 105
8.6.2 Symmetrisches Kartenverbindungsobjekt 107
8.6.3 Privates Schlüsselobjekt 110
8.6.4 Öffentliches Schlüsselobjekt 115
8.6.4.1 Öffentliches Signaturprüfobjekt *116*
8.6.4.2 Öffentliches Authentisierungsobjekt *117*
8.6.4.3 Öffentliches Verschlüsselungsobjekt, Option_Kryptobox *118*
8.7 Datenobjekte (informativ) 119
8.8 Security Environment (informativ) 119
8.9 Sicherheitsstatus (informativ) 121
9 Objektsystem (normativ) 122
9.1 Aufbau und Strukturtiefe 122
9.2 Objektsuche 126
9.2.1 Filesuche 126
9.2.2 Suche nach einem Passwortobjekt 126
9.2.3 Suche nach einem Schlüsselobjekt 128
9.2.3.1 Suche nach einem geheimen Schlüsselobjekt *129*
9.2.3.2 Suche nach einem öffentlichen Schlüsselobjekt *130*

9.3 Cache für öffentliche Schlüsselobjekte 132

10 Zugriffskontrolle (normativ) 136

10.1 Zugriffsart 136

10.2 Zugriffsbedingung 137

10.3 Zugriffsregel 140

10.4 Zugriffsregelauswertung 141

10.5 Auslesen von Zugriffsregeln 142

11 Kommunikation (normativ) 144

11.1 Request - Response 144

11.2 Elektrische Schnittstellen 144
11.2.1 Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11] 144
11.2.2 Übertragungsprotokoll gemäß [ISO/IEC 7816-12] 147
11.2.3 Kontaktlose Datenübertragung gemäß ISO/IEC 14443 148

11.3 OSI-Referenzmodell (informativ) 149

11.4 Kommandobearbeitung 150

11.5 Kommando-APDU 151
11.5.1 Class Byte 151
11.5.2 Instruction Byte 152
11.5.3 Parameter P1 153
11.5.4 Parameter P2 153
11.5.5 Datenfeld 153
11.5.6 LeFeld 154

11.6 Antwort-APDU 154
11.6.1 Datenfeld 155
11.6.2 Trailer 155

11.7 Zulässige Kommando-Antwort-Paare 155
11.7.1 Case 1 Kommando-Antwort-Paar 155
11.7.2 Case 2 Kommando-Antwort-Paar 156
11.7.2.1 Case 2 Short Kommando 156
11.7.2.2 Case 2 Extended Kommando 157
11.7.2.3 Case 2 Response 157
11.7.3 Case 3 Kommando-Antwort-Paare 158
11.7.3.1 Case 3 Short Kommando 158
11.7.3.2 Case 3 Extended Kommando 158
11.7.3.3 Case 3 Response 159
11.7.4 Case 4 Kommando-Antwort-Paare 159
11.7.4.1 Case 4 Short Kommando 160
11.7.4.2 Case 4 Extended Kommando 160
11.7.4.3 Case 4 Response 161

11.8 Command Chaining 162

11.9 Längenbeschränkung von APDU 163

12 Kanalkontext (normativ) 165
12.1 Attribute eines logischen Kanals 165
12.2 Reset-Verhalten 169
12.3 Setzen eines Sicherheitsstatus 170
12.4 Löschen eines Sicherheitsstatus 171
12.4.1 Löschen des Sicherheitszustandes eines Schlüssels 171
12.4.2 Löschen der Sicherheitszustände eines Ordners 172
12.4.3 Löschen von Sessionkeys 173
12.5 Setzen eines Passwortstatus 173
12.6 Löschen eines Passwortstatus 174

13 Gesicherte Kommunikation (normativ) 175
13.1 Secure Messaging Layer 175
13.1.1 Ableitung von Sessionkeys 175
13.1.2 Bearbeitung einer Kommando-APDU 178
13.2 Sicherung einer Kommando-APDU 180
13.3 Sicherung einer Antwort-APDU 183

14 Kommandos (normativ) 186
14.1 Roll-Verhalten 188
14.1.1 Roll-Back 189
14.1.2 Roll-Forward 190
14.2 Management des Objektsystems 190
14.2.1 ACTIVATE 190
14.2.1.1 Use Case Aktivieren eines Ordners oder einer Datei 190
14.2.1.2 Use Case Aktivieren eines privaten oder symmetrischen Schlüsselobjektes 191
14.2.1.3 Use Case Aktivieren eines öffentlichen Schlüsselobjektes 191
14.2.1.4 Use Case Aktivieren eines Passwortobjektes 192
14.2.1.5 Antwort der Karte auf Aktivieren eines Files 193
14.2.1.6 Kommandoabarbeitung innerhalb der Karte 193
14.2.2 CREATE 195
14.2.3 DEACTIVATE 195
14.2.3.1 Use Case Deaktivieren eines Ordners oder einer Datei 195
14.2.3.2 Use Case Deaktivieren eines privaten oder symmetrischen Schlüsselobjektes 196
14.2.3.3 Use Case Deaktivieren eines öffentlichen Schlüsselobjektes 196
14.2.3.4 Use Case Deaktivieren eines Passwortobjektes 197
14.2.3.5 Antwort der Karte auf Deaktivieren eines Files 197
14.2.3.6 Kommandoabarbeitung innerhalb der Karte 198
14.2.4 DELETE 199
14.2.4.1 Use Case Löschen eines Ordners oder einer Datei 200
14.2.4.2 Use Case Löschen eines privaten oder symmetrischen Schlüsselobjektes 200
14.2.4.3 Use Case Löschen eines öffentlichen Schlüsselobjektes 201
14.2.4.4 Use Case Löschen eines Passwortobjektes 201

14.2.4.5 Antwort der Karte auf Löschen eines Files 202
14.2.4.6 Kommandoabarbeitung innerhalb der Karte 202
14.2.5 LOAD APPLICATION 205
14.2.5.1 Use Case Anlegen neues Objekt, nicht Ende der Kommandokette 205
14.2.5.2 Use Case Anlegen neues Objekt, Ende der Kommandokette 206
14.2.5.3 Antwort der Karte auf Anlegen neues Objekt 207
14.2.5.4 Kommandoabarbeitung innerhalb der Karte 207
14.2.6 SELECT 211
14.2.6.1 Use Case Selektieren ohne AID, first, keine Antwortdaten 211
14.2.6.2 Use Case Selektieren ohne AID, first, Antwortdaten mit FCP 211
14.2.6.3 Use Case Selektieren ohne AID, next, keine Antwortdaten 212
14.2.6.4 Use Case Selektieren ohne AID, next, Antwortdaten mit FCP 213
14.2.6.5 Use Case Selektieren per AID, first, keine Antwortdaten 214
14.2.6.6 Use Case Selektieren per AID, first, Antwortdaten mit FCP 214
14.2.6.7 Use Case Selektieren per AID, next, keine Antwortdaten 215
14.2.6.8 Use Case Selektieren per AID, next, Antwortdaten mit FCP 216
14.2.6.9 Use Case Selektieren, DF oder ADF, keine Antwortdaten 217
14.2.6.10 Use Case Selektieren, DF oder ADF, Antwortdaten mit FCP 218
14.2.6.11 Use Case Selektieren übergeordnetes Verzeichnis ohne FCP 218
14.2.6.12 Use Case Selektieren übergeordnetes Verzeichnis mit FCP 219
14.2.6.13 Use Case Selektieren einer Datei, keine Antwortdaten 220
14.2.6.14 Use Case Selektieren einer Datei, Antwortdaten mit FCP 220
14.2.6.15 Use Case Selektieren per AID, first, Antwortdaten mit FCI 221
14.2.6.16 Zusammenfassung der SELECT-Kommando-Varianten 222
14.2.6.17 Antwort der Karte auf Selektieren eines Files 223
14.2.6.18 Kommandoabarbeitung innerhalb der Karte 223
14.2.7 TERMINATE CARD USAGE 227
14.2.7.1 Use Case Terminieren der Karte 227
14.2.7.2 Antwort der Karte auf Terminieren der Karte 228
14.2.7.3 Kommandoabarbeitung innerhalb der Karte 228
14.2.8 TERMINATE DF 229
14.2.8.1 Use Case Terminieren eines Ordners 229
14.2.8.2 Antwort der Karte auf Terminieren eines Ordners 230
14.2.8.3 Kommandoabarbeitung innerhalb der Karte 230
14.2.9 TERMINATE 231
14.2.9.1 Use Case Terminieren einer Datei 231
14.2.9.2 Use Case Terminieren eines privaten oder symmetrischen Schlüsselobjektes 232
14.2.9.3 Use Case Terminieren eines öffentlichen Schlüsselobjektes 232
14.2.9.4 Use Case Terminieren eines Passwortobjektes 233
14.2.9.5 Antwort der Karte auf Terminieren von Datei, Schlüssel- oder Passwortobjekt 233
14.2.9.6 Kommandoabarbeitung innerhalb der Karte 234
14.3 Zugriff auf Daten in transparenten EF 235
14.3.1 ERASE BINARY 236
14.3.1.1 Use Case Löschen ohne shortFileIdentifier in transparenten EF 236
14.3.1.2 Use Case Löschen mit shortFileIdentifier in transparenten EF 237
14.3.1.3 Antwort der Karte auf Löschen in transparenten EF 237
14.3.1.4 Kommandoabarbeitung innerhalb der Karte 238
14.3.2 READ BINARY 240
14.3.2.1 Use Case Lesen ohne shortFileIdentifier in transparenten EF 240

14.3.2.2 Use Case Lesen mit shortFileIdentifier in transparenten EF 240
14.3.2.3 Antwort der Karte auf Lesen in transparenten EF 241
14.3.2.4 Kommandoabarbeitung innerhalb der Karte 242
14.3.3 SEARCH BINARY 244
14.3.4 SET LOGICAL EOF 244
14.3.4.1 Use Case Setzen logical EOF ohne shortFileIdentifier 244
14.3.4.2 Use Case Setzen logical EOF mit shortFileIdentifier 245
14.3.4.3 Antwort der Karte auf Setzen logical EOF in transparenten EF 245
14.3.4.4 Kommandoabarbeitung innerhalb der Karte 246
14.3.5 UPDATE BINARY 248
14.3.5.1 Use Case Schreiben ohne shortFileIdentifier in transparenten EF 248
14.3.5.2 Use Case Schreiben mit shortFileIdentifier in transparenten EF 249
14.3.5.3 Antwort der Karte auf Schreiben in transparenten EF 249
14.3.5.4 Kommandoabarbeitung innerhalb der Karte 250
14.3.6 WRITE BINARY 252
14.3.6.1 Use Case Anfügen ohne shortFileIdentifier in transparenten EF 252
14.3.6.2 Use Case Anfügen mit shortFileIdentifier in transparenten EF 253
14.3.6.3 Antwort der Karte auf Anfügen in transparenten EF 254
14.3.6.4 Kommandoabarbeitung innerhalb der Karte 254

14.4 Zugriff auf strukturierte Daten 256
14.4.1 ACTIVATE RECORD 257
14.4.1.1 Use Case Aktivieren eines Rekords ohne shortFileIdentifier 257
14.4.1.2 Use Case Aktivieren eines Rekords mit shortFileIdentifier 257
14.4.1.3 Use Case Aktivieren aller Rekords ab P1 ohne shortFileIdentifier 258
14.4.1.4 Use Case Aktivieren aller Rekords ab P1 mit shortFileIdentifier 258
14.4.1.5 Antwort der Karte auf Aktivieren von Rekords 259
14.4.1.6 Kommandoabarbeitung innerhalb der Karte 260
14.4.2 APPEND RECORD 262
14.4.2.1 Use Case Anlegen neuer Rekord, ohne shortFileIdentifier 262
14.4.2.2 Use Case Anlegen neuer Rekords, mit shortFileIdentifier 263
14.4.2.3 Antwort der Karte auf Anlegen eines neuen Rekords 263
14.4.2.4 Kommandoabarbeitung innerhalb der Karte 264
14.4.3 DEACTIVATE RECORD 267
14.4.3.1 Use Case Deaktivieren eines Rekords ohne shortFileIdentifier 267
14.4.3.2 Use Case Deaktivieren eines Rekords mit shortFileIdentifier 268
14.4.3.3 Use Case Deaktivieren aller Rekords ab P1 ohne shortFileIdentifier 268
14.4.3.4 Use Case Deaktivieren aller Rekords ab P1 mit shortFileIdentifier 269
14.4.3.5 Antwort der Karte auf Deaktivieren von Rekords 270
14.4.3.6 Kommandoabarbeitung innerhalb der Karte 270
14.4.4 DELETE RECORD 272
14.4.4.1 Use Case Entfernen eines Rekords ohne shortFileIdentifier 272
14.4.4.2 Use Case Entfernen eines Rekords mit shortFileIdentifier 273
14.4.4.3 Antwort der Karte auf Entfernen eines Rekords 274
14.4.4.4 Kommandoabarbeitung innerhalb der Karte 274
14.4.5 ERASE RECORD 276
14.4.5.1 Use Case Löschen eines Rekordinhaltes ohne shortFileIdentifier 276
14.4.5.2 Use Case Löschen eines Rekordinhaltes mit shortFileIdentifier 277
14.4.5.3 Antwort der Karte auf Löschen eines Rekordinhaltes 277
14.4.5.4 Kommandoabarbeitung innerhalb der Karte 278
14.4.6 READ RECORD 280
14.4.6.1 Use Case Lesen ohne shortFileIdentifier in strukturierten EF 280

14.4.6.2 Use Case Lesen mit shortFileIdentifier in strukturierten EF 281
14.4.6.3 Antwort der Karte auf Lesen in strukturierten EF 281
14.4.6.4 Kommandoabarbeitung innerhalb der Karte 282
14.4.7 SEARCH RECORD 284
14.4.7.1 Use Case Suchen ohne shortFileIdentifier in strukturierten EF 284
14.4.7.2 Use Case Suchen mit shortFileIdentifier in strukturierten EF 285
14.4.7.3 Antwort der Karte auf Suchen in strukturierten EF 286
14.4.7.4 Kommandoabarbeitung innerhalb der Karte 286
14.4.8 UPDATE RECORD 288
14.4.8.1 Use Case Rekordinhalt schreiben, ohne shortFileIdentifier 289
14.4.8.2 Use Case Rekordinhalt schreiben, mit shortFileIdentifier 289
14.4.8.3 Antwort der Karte auf Schreiben in strukturierten EF 290
14.4.8.4 Kommandoabarbeitung innerhalb der Karte 291
14.4.9 WRITE RECORD 293

14.5 Zugriff auf Datenobjekte 294
14.5.1 GET DATA 294
14.5.2 PUT DATA 294

14.6 Zugriff auf Passwortobjekte 294
14.6.1 CHANGE REFERENCE DATA 295
14.6.1.1 Use Case Ändern eines Benutzergeheimnisses 295
14.6.1.2 Use Case Setzen eines Benutzergeheimnisses 295
14.6.1.3 Antwort der Karte auf Ändern eines Benutzergeheimnisses 296
14.6.1.4 Kommandoabarbeitung innerhalb der Karte 297
14.6.2 DISABLE VERIFICATION REQUIREMENT 299
14.6.2.1 Use Case Abschalten der Benutzerverifikation mit Benutzergeheimnis 299
14.6.2.2 Use Case Abschalten der Benutzerverifikation ohne Benutzergeheimnis 300
14.6.2.3 Antwort der Karte auf Abschalten der Benutzerverifikation 300
14.6.2.4 Kommandoabarbeitung innerhalb der Karte 301
14.6.3 ENABLE VERIFICATION REQUIREMENT 302
14.6.3.1 Use Case Einschalten der Benutzerverifikation mit Benutzergeheimnis 303
14.6.3.2 Use Case Einschalten der Benutzerverifikation ohne Benutzergeheimnis 303
14.6.3.3 Antwort der Karte auf Einschalten der Benutzerverifikation 304
14.6.3.4 Kommandoabarbeitung innerhalb der Karte 304
14.6.4 GET PIN STATUS 306
14.6.4.1 Use Case Auslesen des Status eines Passwortobjektes 306
14.6.4.2 Antwort der Karte auf Auslesen des PIN Status 307
14.6.4.3 Kommandoabarbeitung innerhalb der Karte 308
14.6.5 RESET RETRY COUNTER 309
14.6.5.1 Use Case Entsperren mit PUK, mit neuem Geheimnis 310
14.6.5.2 Use Case Entsperren mit PUK, ohne neues Geheimnis 310
14.6.5.3 Use Case Entsperren ohne PUK, mit neuem Geheimnis 311
14.6.5.4 Use Case Entsperren ohne PUK, ohne neues Geheimnis 312
14.6.5.5 Antwort der Karte auf Entsperren eines Benutzergeheimnisses 312
14.6.5.6 Kommandoabarbeitung innerhalb der Karte 313
14.6.6 VERIFY 315
14.6.6.1 Use Case Vergleich eines Benutzergeheimnisses 315
14.6.6.2 Antwort der Karte auf Vergleich eines Benutzergeheimnisses 316
14.6.6.3 Kommandoabarbeitung innerhalb der Karte 316

14.7 Komponentenauthentisierung 318
14.7.1 EXTERNAL AUTHENTICATE / MUTUAL AUTHENTICATE 318
14.7.1.1 Use Case Externes Authentisieren ohne Antwortdaten 318
14.7.1.2 Use Case Externes Authentisieren mit Antwortdaten 319
14.7.1.3 Antwort der Karte auf Externes Authentisieren 320
14.7.1.4 Kommandoabarbeitung innerhalb der Karte 321
14.7.2 GENERAL AUTHENTICATE 328
14.7.2.1 Gegenseitiges Authentisieren mittels PACE für Endnutzerkarten 328
14.7.2.1.1 Use Case PACE für Endnutzerkarten, Schritt 1a 328
14.7.2.1.2 Use Case PACE für Endnutzerkarten, Schritt 2a 329
14.7.2.1.3 Use Case PACE für Endnutzerkarten, Schritt 3a 330
14.7.2.1.4 Use Case PACE für Endnutzerkarten, Schritt 4a 330
14.7.2.2 Gegenseitiges Authentisieren mittels ELC Schlüsseln 331
14.7.2.2.1 Use Case gegenseitige ELC-Authentisierung, Schritt 1 331
14.7.2.2.2 Use Case gegenseitige ELC-Authentisierung, Schritt 2 332
14.7.2.3 Authentisieren für asynchrone, symmetrische Kartenadministration 333
14.7.2.3.1 Use Case Authentisieren für asynchrone, sym. Administration, Schritt 1 333
14.7.2.3.2 Use Case Authentisieren für asynchrone, sym. Administration, Schritt 2 333
14.7.2.4 Gegenseitiges Authentisieren mittels PACE für Sicherheitsmodule 334
14.7.2.4.1 Use Case PACE für Sicherheitsmodule, Schritt 1b 334
14.7.2.4.2 Use Case PACE für Sicherheitsmodule, Schritt 2b 335
14.7.2.4.3 Use Case PACE für Sicherheitsmodule, Schritt 3b 336
14.7.2.4.4 Use Case PACE für Sicherheitsmodule, Schritt 4b 336
14.7.2.4.5 Use Case PACE für Sicherheitsmodule, Schritt 5b 337
14.7.2.5 Authentisieren für asynchrone, asym. Kartenadministration 338
14.7.2.5.1 Use Case Authentisieren für asynchrone, asym. Administration, Schritt 1 338
14.7.2.5.2 Use Case Authentisieren für asynchrone, asym. Administration, Schritt 2 338
14.7.2.6 Antwort der Karte auf Generelles Authentisieren 338
14.7.2.7 Kommandoabarbeitung innerhalb der Karte 339
14.7.3 GET SECURITY STATUS KEY 352
14.7.3.1 Use Case Auslesen Sicherheitsstatus symmetrischer Schlüssels, Option_DES 352
14.7.3.2 Use Case Auslesen Sicherheitsstatus einer Rolle, Option_RSA_CVC 353
14.7.3.3 Use Case Auslesen Sicherheitsstatus einer Bitliste 354
14.7.3.4 Antwort der Karte auf Auslesen Sicherheitsstatus eines Schlüssels 354
14.7.3.5 Kommandoabarbeitung innerhalb der Karte 355
14.7.4 INTERNAL AUTHENTICATE 356
14.7.4.1 Use Case internes Authentisieren 356
14.7.4.2 Antwort der Karte auf internes Authentisieren 357
14.7.4.3 Kommandoabarbeitung innerhalb der Karte 358

14.8 Kryptographische Operationen 362
14.8.1 PSO Compute Cryptographic Checksum 363
14.8.1.1 Use Case Berechnen einer kryptographischen Checksumme 363
14.8.1.2 Antwort der Karte auf Berechnen einer kryptographischen Checksumme 364
14.8.1.3 Kommandoabarbeitung innerhalb der Karte 364
14.8.2 PSO Compute Digital Signature 366
14.8.2.1 Use Case Signieren des Datenfeldes, ohne "message recovery" 366
14.8.2.2 Use Case Signieren des Datenfeldes, mit "message recovery" 367
14.8.2.3 Antwort der Karte auf Signieren von Daten 368
14.8.2.4 Kommandoabarbeitung innerhalb der Karte 368
14.8.3 PSO Decipher 370
14.8.3.1 Use Case Entschlüsseln mittels RSA 370
14.8.3.2 Use Case Entschlüsseln mittels ELC 371
14.8.3.3 Use Case Entschlüsseln mittels symmetrischer Schlüssel 372
14.8.3.4 Antwort der Karte auf Entschlüsseln von Daten 373
14.8.3.5 Kommandoabarbeitung innerhalb der Karte 373
14.8.4 PSO Encipher 375
14.8.4.1 Use Case Verschlüsseln von Daten mittels übergebenem RSA-Schlüssel 376
14.8.4.2 Use Case Verschlüsseln von Daten mittels übergebenem ELC-Schlüssel 377
14.8.4.3 Use Case Verschlüsseln mittels gespeichertem RSA-Schlüssel 378
14.8.4.4 Use Case Verschlüsseln mittels gespeichertem ELC-Schlüssel 379
14.8.4.5 Use Case Verschlüsseln mittels symmetrischem Schlüssel 380
14.8.4.6 Antwort der Karte auf Verschlüsseln von Daten 380
14.8.4.7 Kommandoabarbeitung innerhalb der Karte 381
14.8.5 PSO Hash 384
14.8.6 PSO Transcipher 384
14.8.6.1 Use Case Umschlüsseln von Daten mittels RSA-Schlüssel 385
14.8.6.2 Use Case Umschlüsseln von Daten von RSA-Schlüssel nach ELC-Schlüssel 386
14.8.6.3 Use Case Umschlüsseln von Daten mittels ELC 387
14.8.6.4 Use Case Umschlüsseln von Daten von ELC-Schlüssel nach RSA-Schlüssel 388
14.8.6.5 Antwort der Karte auf Umschlüsseln von Daten 389
14.8.6.6 Kommandoabarbeitung innerhalb der Karte 390
14.8.7 PSO Verify Certificate 392
14.8.7.1 Use Case Importieren RSA-Schlüssels mittels Zertifikat, Option_RSA_CVC 393
14.8.7.2 Use Case Importieren eines ELC-Schlüssels mittels Zertifikat 394
14.8.7.3 Antwort den Importieren eines CV-Zertifikates 395
14.8.7.4 Kommandoabarbeitung innerhalb der Karte 395
14.8.8 PSO Verify Cryptographic Checksum 401
14.8.8.1 Use Case Prüfen einer kryptographischen Checksumme 401
14.8.8.2 Antwort der Karte auf Prüfen einer kryptographischen Checksumme ... 402
14.8.8.3 Kommandoabarbeitung innerhalb der Karte 402
14.8.9 PSO Verify Digital Signature 403
14.8.9.1 Use Case Prüfen einer ELC-Signatur 404
14.8.9.2 Antwort der Karte auf Prüfen einer digitalen Signatur 405
14.8.9.3 Kommandoabarbeitung innerhalb der Karte 405

14.9 Verschiedenes 406
14.9.1 ENVELOPE 406
14.9.2 FINGERPRINT 406
14.9.2.1 Use Case Berechnen eines COS-Fingerprints 406
14.9.2.2 Antwort der Karte auf Berechnen eines COS-Fingerprints 407
14.9.2.3 Kommandoabarbeitung innerhalb der Karte 407
14.9.3 GENERATE ASYMMETRIC KEY PAIR 408
14.9.3.1 Use Case Generieren, ohne Überschreiben, ohne Referenz, ohne Ausgabe 408
14.9.3.2 Use Case Generieren, ohne Überschreiben, mit Referenz, ohne Ausgabe 409
14.9.3.3 Use Case Generieren, ggf. Überschreiben, ohne Referenz, ohne Ausgabe 409
14.9.3.4 Use Case Generieren, ggf. Überschreiben, mit Referenz, ohne Ausgabe 410
14.9.3.5 Use Case Auslesen vorhandener Schlüssel, ohne Referenz 411
14.9.3.6 Use Case Auslesen vorhandener Schlüssel, mit Referenz 411
14.9.3.7 Use Case Generieren, ohne Überschreiben, ohne Referenz, mit Ausgabe 412
14.9.3.8 Use Case Generieren, ohne Überschreiben, mit Referenz, mit Ausgabe 413
14.9.3.9 Use Case Generieren, ggf. Überschreiben, ohne Referenz, mit Ausgabe 413
14.9.3.10 Use Case Generieren, ggf. Überschreiben, mit Referenz, mit Ausgabe 414
14.9.3.11 Zusammenfassung der GENERATE ASYMMETRIC KEY PAIR-Kommando-Varianten 415
14.9.3.12 Antwort der Karte auf Generieren oder Auslesen eines asym. Schlüssels 415
14.9.3.13 Kommandoabarbeitung innerhalb der Karte 416
14.9.4 GET CHALLENGE 418
14.9.4.1 Use Case Erzeugen einer Zufallszahl für DES oder RSA Authentisierung 419
14.9.4.2 Use Case Erzeugen einer Zufallszahl für AES oder ELC Authentisierung 419
14.9.4.3 Antwort der Karte auf Erzeugen einer Zufallszahl 420
14.9.4.4 Kommandoabarbeitung innerhalb der Karte 420
14.9.5 GET RANDOM 421
14.9.5.1 Use Case Erzeugen kryptographisch sicherer Zufallszahl 421
14.9.5.2 Antwort der Karte Erzeugen kryptographisch sichere Zufallszahl 421
14.9.5.3 Kommandoabarbeitung innerhalb der Karte 422
14.9.6 GET RESPONSE 423
14.9.7 LIST PUBLIC KEY 423
14.9.7.1 Use Case Auslesen der Liste öffentlicher Schlüsselobjekte 423
14.9.7.2 Antwort der Karte auf Auslesen einer Schlüsselliste 423
14.9.7.3 Kommandoabarbeitung innerhalb der Karte 424
14.9.8 MANAGE CHANNEL 426
14.9.8.1 Use Case Öffnen eines logischen Kanals 426
14.9.8.2 Use Case Schließen eines logischen Kanals 427
14.9.8.3 Use Case Zurücksetzen eines logischen Kanals 427
14.9.8.4 Use Case logischer Reset der Applikationsebene 428

14.9.8.5 Antwort der Karte auf Kanalmanagementoperationen 428
14.9.8.6 Kommandoabarbeitung innerhalb der Karte 429
14.9.9 MANAGE SECURITY ENVIRONMENT 430
14.9.9.1 Use Case Ändern des SE-Identifiers 431
14.9.9.2 Use Case Schlüsselauswahl zur internen, symmetrischen Authentisierung 431
14.9.9.3 Use Case Schlüsselauswahl zur internen, asymmetrischen Authentisierung 432
14.9.9.4 Use Case Schlüsselauswahl zur externen, symmetrischen Authentisierung 433
14.9.9.5 Use Case Schlüsselauswahl zur externen, asymmetrischen Authentisierung 434
14.9.9.6 Use Case Schlüsselauswahl zur symmetrischen, gegenseitigen Authentisierung 435
14.9.9.7 Use Case Schlüsselauswahl zur sym. Kartenverbindung ohne Kurvenangabe 436
14.9.9.8 Use Case Schlüsselauswahl zur sym. Kartenverbindung mit Kurvenangabe 437
14.9.9.9 Use Case Schlüsselauswahl für Signierschlüssel 438
14.9.9.10 Use Case Schlüsselauswahl zum Prüfen von CV–Zertifikaten 439
14.9.9.11 Use Case Schlüsselauswahl zur Datenent- oder Datenumschlüsselung 440
14.9.9.12 Use Case Schlüsselauswahl für Verschlüsselung 441
14.9.9.13 Antwort der Karte auf Management des Security Environments 442
14.9.9.14 Kommandoabarbeitung innerhalb der Karte 442

15 Authentisierungsprotokolle (normativ) 446

15.1 Externe Authentisierung 447
15.1.1 Externe Authentisierung mittels symmetrischer Schlüssel 447
15.1.2 RSA, asymmetrische Rollenauthentisierung, Option_RSA_CVC 448
15.1.3 ELC, asymmetrischer Berechtigungsnachweis 448

15.2 Interne Authentisierung 448

15.3 Card-2-Card-Authentisierung ohne Sessionkey-Aushandlung 449

15.4 Aushandlung von Sessionkey 449
15.4.1 Sessionkeys mittels symmetrischer Authentisierungsobjekte 449
15.4.2 Sessionkeys mittels symmetrischer Kartenverbindungsobjekte 450
15.4.3 Sessionkeyaushandlung mittels RSA-Schlüssel, Option_DES 452
15.4.4 Sessionkeys mittels ELC-Schlüssel 454

15.5 Statisches Secure Messaging 457

16 Verschiedenes (normativ) 459

16.1 Identifier 459

16.2 Codierungen für Trailer 463

17 Hinweise zur Sicherheitsevaluierung (informativ) 467

18 Vorgaben zur Performanz 468

18.1 Einführung (informativ) 468
18.2 Messaufbau (normativ) 468
18.3 Anforderungen an die Steuersoftware (normativ) 469
18.4 Anforderungen an das Interface Device (IFD) (normativ) 469
18.4.1 Anforderungen an das IFD bezüglich T=1 469
18.4.2 Anforderungen an das IFD für [ISO/IEC 7816-12] Datenübertragung 470
18.4.3 Anforderungen an das IFD bezüglich kontaktloser Datenübertragung 470
18.5 Allgemeines (normativ) 471
18.5.1 Normale Zeitmessung 471
18.5.1.1 Normale Zeitmessung für das Übertragungsprotokoll T=1 471
18.5.1.2 Normale Zeitmessung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12] 471
18.5.1.3 Normale Zeitmessung für die kontaktlose Datenübertragung 472
18.5.2 Reguläre Aktivierung der Smartcard 472
18.5.2.1 Reguläre Aktivierung für das Übertragungsprotokoll T=1 472
18.5.2.2 Reguläre Aktivierung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12] 472
18.5.2.3 Reguläre Aktivierung für die kontaktlose Datenübertragung 472
18.5.3 Punkteermittlung 473
18.5.4 Gesamtbewertung 474
18.6 Übertragungsgeschwindigkeit 482
18.6.1 Übertragungsgeschwindigkeit für das Übertragungsprotokoll T=1 482
18.6.2 Übertragungsgeschwindigkeit für das Protokoll [ISO/IEC 7816-12] 482
18.6.3 Übertragungsgeschwindigkeit für kontaktlose Datenübertragung 483
18.7 Startsequenz für das Übertragungsprotokoll T=1 483
18.8 Messverfahren für Einzelkommandos (normativ) 484
18.8.1 ACTIVATE, DEACTIVATE, DELETE, LOAD APPLICATION, TERMINATE 484
18.8.2 SELECT Datei 490
18.8.3 FINGERPRINT 491
18.8.4 TERMINATE CARD USAGE 491
18.8.5 SET LOGICAL EOF, WRITE BINARY 492
18.8.6 ERASE BINARY, UPDATE BINARY 493
18.8.7 READ BINARY 495
18.8.8 Rekord orientierte Kommandos 496
18.8.9 SEARCH RECORD 498
18.8.10 Symmetrische Sessionkeyaushandlung für Secure Messaging 499
18.8.11 Schlüsselimport und asymmetrische Authentisierungsprotokolle 500
18.8.11.1 ELC 256 500
18.8.11.2 ELC 384 503
18.8.11.3 ELC 512 505
18.8.11.4 RSA 2048 509
18.8.11.5 Testablauf Schlüsselimport und asymmetrische Authentisierung 509
18.8.12 INTERNAL AUTHENTICATE zur Rollenauthentisierung 512
18.8.13 PSO Compute Digital Signature mittels signPSS 514
18.8.14 Signaturerzeugung und -verifikation mittels signECDSA 515
18.8.15 PSO Encipher und PSO Decipher mittels rsaDecipherOaep 516
18.8.16 PSO Encipher und PSO Decipher mittels elcSharedSecretCalculation 518
18.8.17 Selektieren von Ordnern und Logical Channel Reset 520

18.8.18 MANAGE SECURITY ENVIRONMENT 520
18.8.19 GENERAL AUTHENTICATE, PACE 522
18.8.20 Symmetrische Sessionkeyaushandlung für Trusted Channel 522
18.8.21 Sessionkeynutzung im Trusted Channel 524
18.8.22 GET RANDOM 526
18.8.23 Öffnen und Schließen logischer Kanäle 527

18.9 Kartenkonfiguration für Performanztests (normativ) 528
18.9.1 Attribute des Objektsystems 529
18.9.1.1 Answer To Reset 529
18.9.2 Allgemeine Festlegungen zu Attributstabellen 530
18.9.3 Root, die Wurzelapplikation 531
18.9.3.1 / MF / EF.ATR 532
18.9.3.2 / MF / EF.DIR 534
18.9.3.3 / MF / EF.GDO 535
18.9.4 Anwendung für Authentisierungsprotokolle, DF.Auth 536
18.9.4.1 / MF/ DF.Auth / PrK.Auth_ELC256 536
18.9.4.2 / MF/ DF.Auth / PrK.Auth_ELC384 537
18.9.4.3 / MF/ DF.Auth / PrK.Auth_ELC512 537
18.9.4.4 / MF/ DF.Auth / PrK.Auth_RSA2048 538
18.9.4.5 / MF/ DF.Auth / PuK.RCA_ELC256 538
18.9.4.6 / MF/ DF.Auth / PuK.RCA_ELC384 539
18.9.4.7 / MF/ DF.Auth / PuK.RCA_ELC512 539
18.9.4.8 / MF/ DF.Auth / PuK.RCA_RSA2048 540
18.9.4.9 / MF / DF.Auth / SK.AES128 540
18.9.4.10 / MF / DF.Auth / SK.AES192 541
18.9.4.11 / MF / DF.Auth / SK.AES256 541
18.9.4.12 / MF / DF.Auth / TC.AES128 542
18.9.4.13 / MF / DF.Auth / TC.AES192 543
18.9.4.14 / MF / DF.Auth / TC.AES256 543
18.9.5 Anwendung für IAS Services, DF.IAS 544
18.9.5.1 / MF/ DF.IAS / PrK.X509_ELC256 545
18.9.5.2 / MF/ DF.IAS / PrK.X509_ELC384 545
18.9.5.3 / MF/ DF.IAS / PrK.X509_ELC512 546
18.9.5.4 / MF/ DF.IAS / PrK.X509_RSA2048 546
18.9.5.5 / MF/ DF.IAS / PrK.X509_RSA3072 548
18.9.6 Anwendung für LCS Use Cases, DF.LCS 550
18.9.6.1 / MF/ DF.LCS / CAN_256 551
18.9.6.2 / MF / DF.LCS / EF.LCS 551
18.9.6.3 / MF / DF.LCS / PIN.LCS 552
18.9.6.4 / MF / DF.LCS / PrK.LCS 553
18.9.6.5 / MF / DF.LCS / PuK.LCS 554
18.9.6.6 / MF / DF.LCS / SK.LCS 554
18.9.7 Anwendung für SELECT EF Use Cases, DF.SelectEF 555
18.9.7.1 / MF / DF.SelectEF / EF.xx 556
18.9.8 Anwendung für rekordorientierte Dateioperationen, DF.strukturiert 556
18.9.8.1 / MF / DF.strukturiert / EF.strukturiert 557
18.9.9 Anwendung für transparente Dateioperationen, DF.transparent 558
18.9.9.1 / MF / DF.transparent / EF.transparent 559

19 Domainparameter elliptischer Kurven (informativ) 560

19.1 ansix9p256r1 ..... **560**
19.2 ansix9p384r1 ..... **560**
19.3 brainpoolP256r1 ..... **561**
19.4 brainpoolP384r1 ..... **562**
19.5 brainpoolP512r1 ..... **562**
20 Erläuterungen zu Wertebereichen (informativ) **564**
21 Verzeichnisse (informativ) **565**
21.1 Abkürzungen **565**
21.2 Glossar **566**
21.3 Abbildungsverzeichnis **566**
21.4 Tabellenverzeichnis **567**
21.5 Referenzierte Dokumente **580**
21.5.1 Dokumente der gematik 580
21.5.2 Weitere Dokumente 580
22 Asynchrone, gesicherte APDU-Sequenzen (informativ) **584**
22.1 Einleitung **584**
22.2 CMS-Kommunikationsmuster **584**
22.2.1 Synchrone Kommunikation zwischen CMS und Smartcard 584
22.2.2 Asynchrone Kommunikation zwischen CMS und Smartcard 584
22.3 Anforderungen an die asynchrone Kommunikation **585**
22.4 Lösungskonzept **586**
22.5 Kryptographie **588**
22.5.1 CMS Aktivitäten 589
22.5.2 Puffer-Aktivitäten 590
22.5.3 Smartcard-Aktivitäten 590
22.6 Auswirkungen auf die Kommandoschnittstelle **591**
23 Speichern öffentlicher Schlüssel (informativ) **592**
23.1 Definitionen **592**
23.2 Perspektiven **593**

1 Einordnung des Dokuments (informativ)

1.1 Zielsetzung

Diese Spezifikation definiert die Anforderungen an die Funktionalität einer Betriebssystemplattform (COS-Plattform) für elektronische Karten im Gesundheitswesen (eGK, HBA, …), die internationalen Standards entsprechen und die internationale sowie europäische Interoperabilität sicherstellen.

Im Einzelnen werden auf der Basis von [ISO/IEC 7816-4], [ISO/IEC 7816-8] und [ISO/IEC 7816-9] Kommandos und Optionen beschrieben, die vom COS zu unterstützen sind.

Der Aufbau dieses Dokumentes gliedert sich wie folgt:

- Kapitel 1: Einordnung des Dokuments enthält Aussagen zum Umgang mit diesem Dokument.
- Kapitel 2: Optionen beschreibt und listet Funktionspakete, die nicht zwingend in allen Kartentypen vorhanden sind.~~wird in einer späteren Version Eingangsanforderungen enthalten, welche die Basis für die normativen Aussagen in späteren Kapiteln bilden werden.~~
- Kapitel 3: Systemüberblick wird in einer späteren Version Grundlagen zu Betriebssystemen für Smartcards enthalten, die auf der Normenreihe ISO/IEC 7816 basieren.
- Kapitel 4: Lebenszyklus von Karte und Applikation grenzt den Gültigkeitsbereich der Spezifikation aus zeitlicher Sicht ab.
- Kapitel 5: Datentypen und Datenkonvertierung definiert einige grundlegende Datentypen, welche die normativen Beschreibungen in späteren Kapiteln vereinfachen.
- Kapitel 6: Kryptographische Algorithmen definiert einige grundlegende kryptographische Funktionen, welche die normativen Beschreibungen in späteren Kapiteln vereinfachen.
- Kapitel 7: CV-Zertifikat beschreibt Zertifikate, welche zu nutzen sind, um öffentliche Schlüssel in eine Smartcard zu importieren.
- Kapitel 8: Objekte enthält eine Art Klassendiagramm in textueller Form. Die dort definierten Objekte und Attribute vereinfachen die normativen Beschreibungen in nachfolgenden Kapiteln.
- Kapitel 9: Objektsystem beschreibt, wie sich Informationen persistent auf einer Smartcard hierarchisch speichern lassen.
- Kapitel 10: Zugriffskontrolle beschreibt den Schutz von Informationen auf einer Smartcard vor unberechtigtem Zugriff.
- Kapitel 11: Kommunikation beschreibt, wie die Smartcard Informationen mit anderen Systemen austauscht.

- Kapitel 12: Kanalkontext beschreibt die Informationen, welche volatil und kanalspezifisch (zu logischen Kanälen siehe auch [ISO/IEC 7816-4]) von der Smartcard gespeichert werden.
- Kapitel 13: Gesicherte Kommunikation beschreibt, wie die Smartcard Informationen kryptographisch geschützt mit anderen Systemen austauscht.
- Kapitel 14: Kommandos enthält normative Aussagen zu Kommandos, welche an eine Smartcard geschickt werden und normative Aussagen, wie diese Kommandos von der Smartcard zu bearbeiten sind. Im Wesentlichen ist dies das wichtigste Kapitel des Dokumentes, weil es die äußere Sichtweise auf das Verhalten der Smartcard an der elektrischen Schnittstelle am umfassendsten beschreibt.
- Kapitel 15: Authentisierungsprotokolle beschreibt Sequenzen, die aus mehr als einem Kommando bestehen.
- Kapitel 16: Verschiedenes spezifiziert konkrete Werte für eine Reihe von Platzhaltern.
- Kapitel 17: Informative Hinweise zur Sicherheitsevaluierung.
- Kapitel 18: Vorgaben zur Performanz enthält Anforderungen an einen Performanztest.

Das Dokument ist "bottom up" aufgebaut, das bedeutet, Artefakte werden zunächst beschrieben und definiert, bevor sie verwendet werden. Für eine "top down" Herangehensweise empfiehlt es sich, mitKapitel 14 zu beginnen. Dort werden, wenn möglich, Verweise auf andere Kapitel gesetzt, wenn Dinge dort ausführlicher beschrieben werden. Wegen der besonderen Bedeutung von Kapitel 14 wird dessen Aufbau im Folgenden näher beleuchtet:

Kapitel 14 enthält alle in der Normenreihe ISO/IEC 7816 standardisierten Kommandos. Der besseren Übersichtlichkeit halber ist Kapitel 14 unterteilt in die Abschnitte Management des Objektsystems, Zugriff auf Daten in transparenten EF, Zugriff auf strukturierte Daten, Benutzerverifikation, Komponentenauthentisierung, kryptographische Operationen und Verschiedenes. Jeder Abschnitt enthält eine Reihe von Unterabschnitten mit Kommandos in alphabetischer Reihenfolge.

1.2 Zielgruppe

Das Dokument richtet sich an Hersteller von Smartcard-Betriebssystemen und an Hersteller von Anwendungen, welche unmittelbar mit einer Smartcard kommunizieren.

1.3 Geltungsbereich

Dieses Dokument enthält normative Festlegungen zur Telematikinfrastruktur des deutschen Gesundheitswesens. Der Gültigkeitszeitraum der vorliegenden Version und deren Anwendung in Zulassungs- oder Abnahmeverfahren wird durch die gematik GmbH in gesonderten Dokumenten (z.B. Dokumentenlandkarte, Produkttypsteckbrief, Leistungsbeschreibung) festgelegt und bekannt gegeben.

Der Inhalt des Dokumentes ist verbindlich für die Erstellung elektronischer Karten im Gesundheitswesen.

Schutzrechts-/Patentrechtshinweis

Die nachfolgende Spezifikation ist von der gematik allein unter technischen Gesichtspunkten erstellt worden. Im Einzelfall kann nicht ausgeschlossen werden, dass die Implementierung der Spezifikation in technische Schutzrechte Dritter eingreift. Es ist allein Sache des Anbieters oder Herstellers, durch geeignete Maßnahmen dafür Sorge zu tragen, dass von ihm aufgrund der Spezifikation angebotene Produkte und/oder Leistungen nicht gegen Schutzrechte Dritter verstoßen und sich ggf. die erforderlichen Erlaubnisse/Lizenzen von den betroffenen Schutzrechtsinhabern einzuholen. Die gematik GmbH übernimmt insofern keinerlei Gewährleistungen.

1.4 Abgrenzung des Dokuments

Dieses Dokument spezifiziert das Verhalten an der elektrischen Schnittstelle zu einem Smartcard-Betriebssystem (Card Operating System, COS). Dieses Dokument spezifiziert NICHT die Architektur des COS. Der einfacheren Darstellung wegen, wird in diesem Dokument von einer modularen Aufteilung des COS ausgegangen. Die hier beschriebene Aufteilung ist nicht verpflichtend. Es wird aber empfohlen, sich an dieser Aufteilung zu orientieren, weil bei künftigen Ergänzungen und Erweiterungen die hier beschriebene Aufteilung zu Grunde gelegt wird.

Die Konfiguration einer Smartcard, also die Festlegung, welche Applikationen, Ordner, Dateien, Schlüssel und Passwörter etwa auf einer Versicherten-, Leistungserbringer-, Geräte- oder sonstigen Karte zu finden sind, ist nicht Gegenstand dieses Dokumentes. Diese finden sich in den kartenspezifischen Festlegungen zum Objektsystem (z. B. in [gemSpec_eGK_ObjSys], [gemSpec_HBA_ObjSys], [gemSpec_SMC-B_ObjSys]).

In Absprache mit den Verantwortlichen des Dokumentes [gemSpec_Krypt] werden in diesem Dokument bewusst Redundanzen zum vorgenannten Dokument akzeptiert. Trotzdem ist [gemSpec_Krypt] relevant für eine konkrete Karte, da dort, anders als in diesem Dokument, normative Vorgaben für die Nutzungsdauer gewisser kryptographischer Verfahren getroffen werden, die hier beschrieben werden.

1.5 Methodik

1.5.1 Nomenklatur der Präfixe

Tabelle 1: CosT_1e0: Präfixe, die auf Vielfachen von Zehnerpotenzen beruhen

Name	Symbol	Wert gemäß SI	nächstliegende Zweierpotenz
kilo	k	10^3 = 1.000	2^{10} = 1.024
mega	M	10^6 = 1.000.000	2^{20} = 1.048.576
giga	G	10^9 = 1.000.000.000	2^{30} = 1.073.741.824
tera	T	10^{12} = 1.000.000.000.000	2^{40} = 1.099.511.627.776
peta	P	10^{15} = 1.000.000.000.000.000	2^{50} = 1.125.899.906.842.624
exa	E	10^{18}	2^{60}
zetta	Z	10^{21}	2^{70}
yotta	Y	10^{24}	2^{80}

Die folgende Tabelle basiert auf [BinPrefix].

Tabelle 2: CosT_c5e: Präfixe, die auf Vielfachen von Zweierpotenzen beruhen

Name	Symbol	Wert
kibi	Ki	2^{10} = 1024^1 = 1.024
mebi	Mi	2^{20} = 1024^2 = 1.048.576
gibi	Gi	2^{30} = 1024^3 = 1.073.741.824
tebi	Ti	2^{40} = 1024^4 = 1.099.511.627.776
pebi	Pi	2^{50} = 1024^5 = 1.125.899.906.842.624
exbi	Ei	2^{60} = 1024^6 = 1.152.921.504.606.846.976
zebi	Zi	2^{70} = 1024^7 = 1.180.591.620.717.411.303.424
yobi	Yi	2^{80} = 1024^8 = 1.208.925.819.614.629.174.706.176

Hinweis CosH_3d9: Beispiel: 300 GB ≈ 279,4 GiB, sprich 300 Gigabyte sind ungefähr 279,4 Gibibyte.

1.5.2 Nomenklatur Verschiedenes

In diesem Dokument wird eine objektorientierte Sichtweise verfolgt. Dazu werden etwa die Artefakte Datei (EF in der Nomenklatur nach [ISO/IEC 7816-4]) oder Schlüssel als Objekte aufgefasst und die Eigenschaften als Attribute des Objektes. Wenn Attribute eines Objektes angesprochen werden, dann wird die Notation *obj.attribute* verwendet. Wenn das Attribut wieder ein Objekt mit weiteren Attributen ist, dann sind auch längere Bezeichnungen möglich.

Tabelle 3: CosT_9f7: Abkürzungen und Definitionen

~~G1~~	~~Abkürzung für Generation 1, bezeichnet ältere Versionen des Dokumentes~~
G2	Abkürzung für Generation 2, bezeichnet aktuelle Versionen des Dokumentes
'1D'	Hexadezimale Zahlen und Oktettstrings werden in Hochkommata eingeschlossen
'XX'	Ein Oktett mit beliebigem Inhalt. Obwohl für das obere und untere Nibble dasselbe Symbol verwendet wird, ist es möglich, dass die Nibble unterschiedlich sind.
'XX…XX'	Ein Oktettstring beliebiger Länge und beliebigen Inhalts.
x || y	Das Symbol || steht für die Konkatenierung von Oktettstrings oder Bitstrings '1234' || '5678' == '12345678'
$y = x$	Der Variablen y wird der Wert von x zugewiesen (Standardnotation in gängigen Programmiersprachen).

Diese Dokumentenversion wurde mit einem neuen Tool statt mit Word erstellt. Das hat folgende Auswirkungen:

1. In der Normenreihe ISO/IEC 7816 ist es üblich Kommandonamen mit Kapitälchen darzustellen. Weil dieses Feature im Tool nicht verfügbar ist, werden Kommandonamen in einem anderen Font dargestellt, beispielsweise READ BINARY, PSO Compute Digital Signature.
2. Die N-Nummern enthalten oft eine Untergliederung in Unterpunkte, Unter-Unterpunkte, etc. Die Nummerierung erfolgt in Word und neuem Tool teilweise unterschiedlich:
    a. Die erste Gliederungsstufe ist sowohl in Word, als auch im Tool alphabetisch: a, b, c, d, ...
        i. Die zweite Gliederungsstufe erfolgt numerisch mit arabischen Ziffern (Word: 1, 2, ...) bzw. kleinen römischen Ziffern (Tool: i, ii, iii, iv, v, ...)
            A. Die dritte Gliederungsstufe erfolgt in Word numerisch mit kleinen römischen Ziffern (i, ii, iii, iv, v, vi, ...) und im Tool alphabetisch mit Großbuchstaben (A, B, C, ...).
                I. Die vierte Gliederungsstufe erfolgt in Word alphabetisch mit Großbuchstaben (A, B, C, ...) und im Tool numerisch mit großen römischen Ziffern (I, II, III, IV, V, VI, ...)
                    1. Weitere Gliederungsebenen werden in diesem Dokument in Anforderungen nicht verwendet.
    b. Daraus folgt, dass Referenzen auf N-Nummern sich von der Word-Version hin zur Tool-Version möglicherweise unterscheiden, aber ineinander umrechenbar sind. Zudem ist erkennbar, falls in der Tool-Version die Umrechnung unterblieb. Beispiel: Eine Referenz auf (N654.321)a.7.ii.D ist eine Word-Version, die im Tool so lautet: (N654.321)a.vii.B.IV. Abgesehen von dieser anderen Art der Darstellung von N-Nummern auf tieferen Gliederungsebenen sind die N-Nummern im Tool identisch zu den N-Nummern in der Word-Version.

3. Automatische Referenzierungen wie in Word (beispielsweise siehe Abbildung 3 oder siehe Tabelle 47) sind im Tool nicht möglich. Ersatzweise werden händisch gepflegte Label nach folgendem Schema verwendet:
   a. CosA_xxx für Abbildungen,
   b. CosH_xxx für Hinweise,
   c. CosT_xxx für Tabellen
   d. wobei xxx eine dreistellige Folge hexadezimaler Ziffern ist.

1.5.3 Normative und informative Abschnitte

Abschnitte mit normativen Inhalten tragen hinter der Kapitelüberschrift den Hinweis:

(normativ)

Generell gilt, dass lediglich die Gliederungen, welche durch eine Nummer (N4711) gekennzeichnet sind, zulassungsrelevante Eigenschaften enthalten ~~SOLLEN~~ und somit im Rahmen der Zulassung getestet werden ~~SOLLEN~~. Falls dies in einem speziellen Fall nicht so ist, handelt es sich höchstwahrscheinlich um einen editorischen Fehler.

1.5.4 Komponentenspezifische Anforderungen

Da es sich beim vorliegenden Dokument um die Spezifikation einer Schnittstelle zwischen mehreren Komponenten handelt, ist es unvermeidlich, dass dieses Dokument Anforderungen für jede der Komponenten enthält. Die normativen Abschnitte tragen deshalb eine Kennzeichnung, auf welche Komponente sich die Anwendung primär bezieht. Dabei gelten natürlicherweise folgende Zusammenhänge:

- Alle normativen Anforderungen an die Komponenten K_Anwendungsspezifikation und K_externeWelt sind auch normative Anforderungen an diejenigen Komponenten, die in den jeweiligen Mengenklammern genannt sind.
- Für eine in der Mengenklammer aufgeführte Komponente ist es zulässig, mehr zu unterstützen als durch K_Anwendungsspezifikation oder K_externeWelt gefordert.
- Für eine in der Mengenklammer aufgeführte Komponente ist es zulässig, die Unterstützung von Dingen abzulehnen, die durch K_Anwendungsspezifikation oder K_externeWelt nicht gefordert werden.

Die obigen Aussagen werden im Folgenden durch Beispiele verdeutlicht:

- In (N007.900) wird von K_Anwendungsspezifikation {K_Karte} gefordert, nur bestimmte *seIdentifier* zu verwenden, wenn die Anwendungsspezifikation für die Komponente K_COS bestimmt ist. Damit ist auch die maximale Anzahl von möglichen Security Environments in der Anwendung beschränkt. Einer Anwendungsspezifikation ist es nicht erlaubt, mehr SEs zu verwenden. Für das COS der Komponente K_COS bedeutet dies, dass es mindestens diese maximale Anzahl an SE zu unterstützen hat. Für das COS der Komponente K_COS ist es sowohl zulässig, mehr SEs zu unterstützen als auch zusätzliche SEs abzulehnen.Kapitel 7

- In 14.3.2.1 und 14.3.2.2 werden im Zusammenhang mit dem Kommando READ BINARY Anforderungen an K_externeWelt {K_Karte} gestellt. Für die externe Welt ist es unzulässig, andere READ BINARY-Varianten zu verwenden. Für das COS bedeutet dies, dass es mindestens diese Varianten zu unterstützen hat. Für das COS ist es sowohl zulässig. mehr READ BINARY-Varianten zu unterstützen als auch zusätzliche READ BINARY-Varianten abzulehnen.

Tabelle 4: CosT_485: Liste der Komponenten, an welche dieses Dokument Anforderungen stellt

Komponente	Beschreibung
K_Anwendungsspezifikation {…}	Instanz, welche eine Anwendung spezifiziert; damit gilt diese Anforderung auch für jede Anwendungsspezifikation, die für eine in der Mengenklammer genannte Komponente bestimmt ist
K_COS	Betriebssystem einer Smartcard
~~K_COS_G1~~	~~Betriebssystem einer Smartcard in der Generation 1, es ist denkbar, dass derartig gekennzeichnete Anforderungen in späteren Versionen dieses Dokumentes entfallen~~
K_IC	das IC einer Smartcard
K_Karte	beliebiger Kartentyp, Oberbegriff für die Menge {eGK, HBA, …}
K_externeWelt {…}	Instanz, welche Nachrichten generiert, um diese an eine in der Mengenklammer genannte Komponente zu senden
K_TST	Komponente funktionaler Zulassungstest

1.5.5 Verwendung von Schlüsselworten

Anforderungen als Ausdruck normativer Festlegungen werden durch eine eindeutige ID in Klammern sowie die dem [RFC2119] entsprechenden, in Großbuchstaben geschriebenen deutschen Schlüsselworte MUSS, DARF NICHT, SOLL, SOLL NICHT, KANN gekennzeichnet. ~~Abwandlungen~~Konjugationen von "MUSS" zu "MÜSSEN" etc. sind der Grammatik geschuldet. Falls in einem speziellen Fall die Verben müssen, dürfen, sollen, können ohne einen einzigen Großbuchstaben verwendet werden, handelt es sich höchstwahrscheinlich um einen editorischen Fehler.

Da im Beispielsatz "*Eine leere Liste DARF NICHT ein Element besitzen*." die Phrase "DARF NICHT" semantisch irreführend wäre (wenn nicht ein, dann vielleicht zwei?), wird stattdessen "*Eine leere Liste DARF KEIN Element besitzen*." verwendet.

In dieser Version wird das Verb "kann" in normativen Festlegungen nur noch für optionale Funktionspakete verwendet, aber nicht länger im Sinne von "zulässigen Implementierungsvarianten" oder "herstellerspezifischen Erweiterungen". Beispiel alt: (N002.200) Das COS KANN weitere Moduluslängen unterstützen oder ablehnen. Die gematik betrachtet normative Festlegungen mit dem Verb "kann" als Implementierungsvarianten und erfragt bei Herstellern welche konkrete Ausprägung gewählt wurde. Das ist im gezeigten Beispiel sinnlos. Deshalb wird die etwa in (N002.200) an einen Hersteller gerichtete Erlaubnis bestimmte Erweiterungen vorzunehmen oder zu unterlassen methodenkonform angepasst.

~~In diesem Dokument werden Aussagen mit dem Schlüsselwort KANN generell sowohl positiv als auch negativ formuliert. (N002.200) ist im Zusammenhang mit (N002.100)a dafür ein gutes Beispiel. In (N002.100)a wird eine normative Forderung erhoben, die offen lässt, ob zusätzliche Werte fakultativ verboten sind oder nicht. Diese Lücke wird durch (N002.200) geschlossen.~~

2 Optionen (normativ)

Dieses Unterkapitel listet Funktionspakete auf, die nicht zwingend erforderlich sind für eine Zulassung des COS im Sinne dieses Dokumentes.

A_13543 - (N000.020) K_IC {K_Karte}, Option_USB_Schnittstelle
a. Das IC und das COS einer Smartcard KÖNNEN die Option_USB_Schnittstelle unterstützen.
b. Wenn das IC und das COS einer Smartcard die Option_USB_Schnittstelle

1. unterstützen, dann sind ~~MÜSSEN~~ zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_USB_Schnittstelle gekennzeichnet sind.
2. nicht unterstützen, dann sind ~~DÜRFEN NICHT~~ mit Option_USB_Schnittstelle gekennzeichnete Anforderung irrelevant für funktionale Tests ~~sein~~.

[<=]

A_13546 - (N000.022) K_IC {K_Karte}, Option_kontaktlose_Schnittstelle
a. Das IC und das COS einer Smartcard KÖNNEN die Option_kontaktlose_Schnittstelle unterstützen.
b. Wenn das IC und das COS einer Smartcard die Option_kontaktlose_Schnittstelle

1. unterstützen, dann sind ~~MÜSSEN~~ zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_kontaktlose_Schnittstelle gekennzeichnet sind.
2. nicht unterstützt, dann sind ~~DÜRFEN NICHT~~ mit Option_kontaktlose_Schnittstelle gekennzeichnete Anforderung irrelevant für funktionale Tests ~~sein~~.

[<=]

A_13549 - (N000.024) K_COS {K_Karte}, Option_logische_Kanäle
a. Das COS KANN die Option_logische_Kanäle unterstützen.
b. Wenn das COS die Option_logische_Kanäle

1. unterstützt, dann sind ~~MÜSSEN~~ zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_logische_Kanäle gekennzeichnet sind.
2. nicht unterstützt, dann sind ~~DÜRFEN NICHT~~ mit Option_logische_Kanäle gekennzeichnete Anforderung irrelevant für funktionale Tests ~~sein~~.

[<=]

A_13552 - (N000.026) K_COS {K_Karte}, Option_Kryptobox
a. Das COS KANN die Option_Kryptobox unterstützen.
b. Wenn das COS die Option_Kryptobox

1. unterstützt, dann sind ~~MÜSSEN~~ zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_Kryptobox gekennzeichnet sind.
2. nicht unterstützt, dann sind ~~DÜRFEN NICHT~~ mit Option_Kryptobox gekennzeichnete Anforderung irrelevant für funktionale Tests ~~sein~~.

[<=]

A_13555 - (N000.028) K_COS {K_Karte}, Option_PACE_PCD
a. Das COS KANN die Option_PACE_PCD unterstützen.
b. Wenn das COS die Option_PACE_PCD

1. unterstützt, dann sind ~~MÜSSEN~~ zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_PACE_PCD gekennzeichnet sind.
2. nicht unterstützt, dann sind ~~DÜRFEN NICHT~~ mit Option_PACE_PCD gekennzeichnete Anforderung irrelevant für funktionale Tests ~~sein~~.

[<=]

Die folgenden Punkte sind absichtlich leer: (N000.030), (N000.032).

A_13558 - (N000.030) K_COS {Karte}, Option_DES
~~a. Das COS KANN die Option_DES unterstützen.~~
~~b. Wenn das COS die Option_DES~~

1. ~~unterstützt, dann sindMÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_DES und Option_RSA_CVC gekennzeichnet sind.~~
2. ~~nicht unterstützt, dann sindDÜRFEN NICHT mit Option_DES gekennzeichnete Anforderung irrelevant für funktionale Tests.~~

[<=]

~~*Hinweis CosH_032: Die Option_DES wird derzeit lediglich zur Implementierung von Karten der Generation 1 verwendet, also eGK G1 und eGK G1+. Die Option_DES ist für Karten der Generation 2 und später irrelevant.*~~

A_13561 - (N000.032) K_COS {Karte}, Option_RSA_CVC
~~a. Das COS KANN die Option_RSA_CVC unterstützen.~~
~~b. Wenn das COS die Option_RSA_CVC~~

1. ~~unterstützt, dann sindMÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_RSA_CVC gekennzeichnet sind.~~
2. ~~nicht unterstützt, dann sindDÜRFEN NICHT mit Option_RSA_CVC gekennzeichnete Anforderung irrelevant für funktionale Tests.~~

[<=]

~~*Hinweis CosH_14c: Die Option_RSA_CVC wird derzeit zur Implementierung folgender Karten benötigt:*~~
~~*a. Karten der Generaion 1, also eGK G1 und eGK G1+,*~~
~~*b.HBA und SMC-B der Generation 2 und Generation G2.1.*~~

A_13564 - (N000.034) K_COS {Karte}, Option_RSA_KeyGeneration
a. Das COS KANN die Option_RSA_KeyGeneration unterstützen.
b. Wenn das COS die Option_RSA_KeyGeneration

1. unterstützt, dann sind zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen zu erfüllen, die mit Option_RSA_KeyGeneration gekennzeichnet sind.

2. nicht unterstützt, dann sind mit Option_RSA_KeyGeneration gekennzeichneten Anforderung irrelevant für funktionale Tests.

[<=]

Hinweis CosH_84b: Die Option_RSA_KeyGeneration wird derzeit lediglich im Rahmen der Personalisierung von Karten der Typen HBA und SMC-B sowie bei der Umsetzung der Option_lange_Lebensdauer verwendet.

3 Systemüberblick (informativ)

1. Smartcards sind sichere Datenspeicher.
   a. Smartcards speichern Daten in Dateien oder Rekords.
   b. Smartcards speichern personenbezogene Schlüssel für IAS-Services.
2. Smartcards kontrollieren den Zugriff mittels Zugriffsregeln.
   a. Ein personenbezogener Zugriff wird durch Passwörter ermöglicht.
   b. Ein rollenbezogener Zugriff wird durch Authentisierungsschlüssel ermöglicht.
   c. Eine sichere Kommunikation wird durch "Trusted Channel" ermöglicht.
3. Eine Smartcard kann sowohl als Start- als auch als Endpunkt eines Trusted Channels fungieren. D. h. diese Smartcard kann einerseits Sessionkeys im Applikationslayer für PSO-Kommandos verwenden, als auch im Secure Messaging Layer. Wo genau hängt von der *algId* bei der gegenseitigen Authentisierung ab.

4 Lebenszyklus von Karte und Applikation (informativ)

In der Literatur finden sich verschiedene Beschreibungen für den Lebenszyklus von Karten. Dieses Kapitel stellt eine vereinfachte Sicht dar und legt dabei einen Gültigkeitsbereich für diese Spezifikation fest. Grob lässt sich der Lebenszyklus einer Karte in drei Phasen einteilen:

1. **Vorbereitungsphase:** Diese Phase umfasst aus Sicht der Produktion alle Schritte, die erforderlich sind, um eine Karte für die Nutzungsphase vorzubereiten. Dazu zählen im Wesentlichen die Entwicklung des Betriebssystems, dessen Test, Abnahme und gegebenenfalls auch Evaluierung und Zulassung. Entsprechende Chips werden anschließend produziert, initialisiert und personalisiert. Die Chips werden in einen Kartenkörper implantiert und an einen Kartennutzer ausgeliefert. Die Reihenfolge der Produktionsschritte weicht unter bestimmten Umständen von der genannten Reihenfolge ab und ist hier lediglich beispielhaft skizziert. Dieses Dokument gilt nicht für die Vorbereitungsphase der Karte. Die Vorbereitungsphase der Karte endet spätestens mit der Übergabe der Karte an einen Kartennutzer. Dann beginnt die Nutzungsphase der Karte.
2. **Nutzungsphase:** Diese Phase umfasst den elektrischen Gebrauch der Karte. Dieses Dokument gilt für die Nutzungsphase der elektrischen Kartenschnittstelle. Die Nutzungsphase der Karte endet, wenn sämtliche Business Use Cases irreversibel gesperrt sind, mithin also auch, wenn die Karte physikalisch zerstört wird.
3. **Terminierungsphase:** Befindet sich die Karte in der Terminierungsphase, dann sind typischerweise alle intendierten Nutzungen der Karte irreversibel gesperrt. In der Regel lassen sich also weder Daten auslesen noch speichern und es ist keine Benutzerverifikation und auch keine Komponentenauthentisierung mehr möglich. Dies ist erreichbar durch eine physikalische Zerstörung des Chips, oder etwa auch durch die Unterstützung des Kommandos TERMINATE CARD USAGE (siehe (N048.700)). Da nach Ausführung eines solchen Kommandos herstellerspezifisch noch gewisse Kommandos möglich sind (etwa SELECT, GET CHALLENGE, …) oder die Übertragungsschicht T=1 möglicherweise noch aktiv ist, ist es nicht möglich, hier von einer Karte zu sprechen, die völlig inaktiv ist.

Analog zu den Phasen einer Karte ist es möglich, auch für Applikationen oder deren Bestandteile (Dateien, Passwörter, Schlüssel, …) die Phasen Vorbereitung, Nutzung und Terminierung zu definieren. Die Aussagen zur physikalischen Zerstörung der Karte gehen dann über in ein Löschen der Applikation oder deren Bestandteile.

Dieses Dokument gilt nicht für die Vorbereitungsphase von Applikationen oder deren Bestandteile.~~Sie beschreibt lediglich den Zustand des Objektsystems in der Nutzungsphase.~~

Die Nutzungsphase einer Applikation oder eines Applikationsbestandteils beginnt, sobald sich ein derartiges Objekt, wie in der Spezifikation der Anwendung definiert, verwenden lässt. Die Nutzungsphase einer Applikation oder eines Applikationsbestandteils endet, wenn das entsprechende Objekt gelöscht wird.

5 Datentypen und Datenkonvertierung (normativ)

Dieses Dokument verwendet die folgenden Datentypen äquivalent zu [BSI-TR-03111#3]:

1. Oktettstring (OS),
2. Bitstring (BS),
3. Integer (I),
4. Körperelement (Field Element FE) und
5. elliptischen Kurvenpunkt (ECP).

Definition: Das höchstwertige Bit (most significant bit, MSBit) eines Bitstrings ist das am weitesten links stehende.

Definition: Das niedrigstwertige Bit (least significant bit, LSBit) eines Bitstrings ist das am weitesten rechts stehende.

Definition: Das höchstwertige Oktett (most significant byte, MSByte) eines Oktettstrings ist das am weitesten links stehende.

Definition: Das niedrigstwertige Oktett (least significant byte, LSByte) eines Oktettstrings ist das am weitesten rechts stehende.

Dieses Dokument verwendet die folgenden Konvertierungsfunktionen äquivalent zum Dokument [BSI-TR-03111#3.1]:

1. Bitstring nach Oktettstring BS2OS,
2. Oktettstring nach Bitstring OS2BS,
3. Körperelement nach Oktettstring FE2OS,
4. Oktettstring nach Körperelement OS2FE.

5.1 BitLength Anzahl Bit in einem Bitstring

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

A_18235 - (N000.080) K_COS

Tabelle 5: CosT_89d: Definition der Funktion BitLength(...)

Input:	*in*	Bitstring mit beliebigem Inhalt und beliebiger Länge
Output:	*out*	Integer, Anzahl der Bits aus denen *in* besteht
Errors:	–	Keine
Notation:		*out* = BitLength(*in*)

Das COS MUSS in *out* die Anzahl Bits von *in* zurückmelden.**[<=]**

Hinweis CosH_fca: Beispiele:
a. BitLength('') = 0
b. BitLength('0') = 1, BitLength('1') = 1,
c. BitLength('00') = 2, BitLength('01') = 2,
d. BitLength('10') = 2, BitLength('11') = 2, …

5.2 OctetLength Anzahl Oktett in einem Oktettstring

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

A_18234 - (N000.090) K_COS

Tabelle 6: CosT_f2b: Definition der Funktion OctetLength(...)

Input:	*in*	Oktettstring mit beliebigem Inhalt und beliebiger Länge, oder nicht–negative ganze Zahl
Output:	*out*	Integer, Anzahl der Oktette aus denen *in* besteht, oder Anzahl Oktette, die mindestens nötig sind, um eine nicht–negative ganze Zahl zu codieren.
Errors:	–	Keine
Notation:		*out* = OctetLength(*in*)

a. Wenn *in* ein Oktettstring ist, dann MUSS das COS in *out* die Anzahl der Oktett in *in* in zurückmelden.
b. Wenn *in* eine nicht-negative ganze Zahl ist, dann MUSS das COS in *out* die Mindestanzahl an Oktetten zurückmelden, die notwendig ist um *in* hexadezimal darzustellen.**[<=]**

Hinweis CosH_f1b: Beispiele:
a. OctetLength('') = 0
b. OctetLength('0034') = 2
c. OctetLength(0) = 1, weil die Zahl 0 in einem Oktett codierbar ist, 0 = '00'.
d. OctetLength(127) = 1, weil 127 = '7F`
e. OctetLength(255) = 1, weil 255 = 'FF'
f. OctetLength(256) = 2, weil 256 = '0100'.

Hinweis CosH_3cb, ACHTUNG: Der Oktettstring '0000FFFF' lässt sich als Repräsentant der Zahl 65535 interpretieren. Ohne führende Nullen lautet die hexadezimale Repräsentation der Zahl 65535='FFFF'. Daraus folgt: OctetLength('0000FFFF') = 4, aber OctetLength(65535) = 2.

5.3 I2OS Integer nach Oktettstring

Dieser Abschnitt beschreibt die Konvertierung einer nicht-negativen ganzen Zahl in einen Oktettstring. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

A_13586 - (N000.100) K_COS

Tabelle 7: CosT_042: Definition der Funktion I2OS(...)

Input:	x	Integer, nicht-negative ganze Zahl
	n	Integer, nicht-negative ganze Zahl, Anzahl der Oktette in out, n darf null sein, dann ist out leer
Output:	out	Oktettstring der Länge n Oktett
Errors:	–	Keine *ACHTUNG: Im Unterschied zu [BSI-TR-03111#3.1.2] wird hier keine Fehlermeldung erzeugt, falls x größer gleich 256^n ist*
Notation:		out = I2OS(x, n)

Das Prinzip ist, die nicht-negative ganze Zahl x als Ziffernfolge zur Basis 256 zu notieren und dann nur die niedrigwertigsten n Ziffern für die Ausgabe zu verwenden. Das COS MUSS dabei folgenden Algorithmus ausführen:

a. Schritt 1: $x = 256^0 x_0 + 256^1 x_1 + 256^2 x_2 + \ldots + 256^i x_i + \ldots$

b. Schritt 2: $M_i = x_i$.

~~Anmerkung~~Hinweis CosH_bfc: Jede Ziffer x_i wird vorzeichenlos in einem Oktett M_i codiert.

c. Schritt 3: $out = M_{n-1} \;||\; M_{n-2} \;||\; \ldots \;||\; M_2 \;||\; M_1 \;||\; M_0$. **[<=]**

Hinweis CosH_7e6: In gewissen Grenzen ist dies die Umkehrfunktion zu (N000.200) (N000.200).

Hinweis CosH_ff7: Beispiele:

a. I2OS(30010, 1) = '3A',
b. I2OS(30010, 2) = '753A',
c. I2OS(30010, 3) = '00753A'.

5.4 OS2I Oktettstring nach Integer

Dieser Abschnitt beschreibt die Konvertierung eines Oktettstrings in eine nicht–negative ganze Zahl. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

A_13587 - (N000.200) K_COS

Tabelle 8: CosT_c6a: Definition der Funktion OS2I(...)

Input:	in	Oktettstring beliebiger Länge und beliebigen Inhalts
Output:	out	Integer, nicht–negative ganze Zahl
Errors:	–	Keine
Notation:		out = OS2I(in)

Das Prinzip ist, jedes Oktett als Ziffer zur Basis 256 einer nicht-negativen ganzen Zahl im Big-Endian-Format aufzufassen. Das COS MUSS dabei folgenden Algorithmus ausführen:

a. Schritt 1: n = OctetLength(in)

b. Wenn n gleich null ist,
   1. dann ist out = 0.
   2. sonst wähle out so, dass I2OS(out, n) identisch zu in ist.**[<=]**

Hinweis CosH_07c: In gewissen Grenzen ist dies die Umkehrfunktion zu (N000.100) (N000.100).

5.5 OS2P Oktettstring nach Punkt (uncompressed encoding)

Dieser Abschnitt beschreibt die Konvertierung eines Oktettstrings in einen Punkt auf einer elliptischen Kurve. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

G2_N000.300 - (N000.300) K_COS

Tabelle 9: CosT_3e0: Definition der Funktion OS2P(...)

Input:	PO	Oktettstring, codiert einen Punkt auf einer elliptischen Kurve
	dP	Domainparameter gemäß (N008.600)
Output:	P	Punkt auf einer elliptischen Kurve mit den Koordinaten P = (x, y)
Errors:	ERROR	Wenn PO nicht im Format "uncompressed encoding" vorliegt
Notation:		P = OS2P(PO, dP)

Das COS MUSS die Decodierung von PO gemäß [BSI-TR-03111#3.2.1] durchführen:
a. Schritt 1: Wenn OctetLength(PO) ungleich (2 $dP.L$ + 1) ist,
   dann gebe den Fehler "ERROR" zurück und beende diesen Algorithmus.
b. Schritt 2: Teile PO auf gemäß:
   PO == PC || X || Y,
   1 == OctetLength(PC),
   $dp.L$ == OctetLength(X) == OctetLength(Y).
c. Schritt 3: Wenn PC ungleich '04' ist,
   dann gebe den Fehler "ERROR" zurück und beende diesen Algorithmus.
d. Schritt 4: Setze P = (x, y) = (OS2I(X) mod $dP.p$, OS2I(Y) mod $dP.p$).
e. Schritt 5: Wenn P nicht auf der durch dP definierten Kurve liegt,
   dann gebe den Fehler "ERROR" zurück und beende diesen Algorithmus.**[<=]**

5.6 P2OS Endlicher Punkt nach Oktettstring

Dieser Abschnitt beschreibt die Konvertierung eines Punktes auf einer elliptischen Kurve in einen Oktettstring. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird diese Konvertierung als Funktion wie folgt verwendet:

A_13588 - (N000.400) K_COS

Tabelle 10: CosT_e43: Definition der Funktion P2OS(...)

Input:	P	Punkt auf einer elliptischen Kurve mit den Koordinaten P = (x, y)
	n	Integer, positive ganze Zahl, Anzahl Oktette pro Koordinate
Output:	PO	Oktettstring, codiert einen Punkt auf einer elliptischen Kurve
Errors:	–	
Notation:		PO = P2OS(P, n)

Das COS MUSS die Codierung von P gemäß [BSI-TR-03111#3.2.1] vornehmen:
PO = '04' || I2OS(x, n) || I2OS(y, n).**[<=]**

5.7 Extrahiere führende Elemente

5.7.1 Extrahiere führende Bits

Dieser Abschnitt beschreibt, wie aus einem Bitstring oder Oktettstring führende Bits extrahiert werden.

A_18230 - (N000.500) K_COS
Das COS MUSS beim Aufruf der Funktion Extract_MSBit(...) sicherstellen, dass n kleiner gleich s ist.**[<=]**

A_18231 - (N000.600) K_COS

Tabelle 11: CosT_638: Definition der Funktion Extract_MSBit(...)

Input:	in	Entweder Bitstring der Länge s Bit, oder Oktettstring der Länge s Bit
	n	Integer, Anzahl der zu extrahierenden Bit
Output:	out	Bitstring der Länge n Bit
Errors:	–	Keine
Notation:		out = Extract_MSBit(in, n)

Das COS MUSS in den Bitstring out die n MSBit von in kopieren.**[<=]**

5.7.2 Extrahiere führende Oktette

Dieser Abschnitt beschreibt, wie aus einem Oktettstring führende Oktette extrahiert werden.

A_18232 - (N000.700) K_COS
Das COS MUSS beim Aufruf der Funktion ExtractMSByte(...) sicherstellen, dass n kleiner gleich s ist.**[<=]**

A_18233 - (N000.800) K_COS

Tabelle 12: CosT_1ff: Definition der Funktion Extract_MSByte(...)

Input:	*in*	Oktettstring der Länge *s* Oktett
	n	Integer, Anzahl der zu extrahierenden Oktette ~~Elemente~~
Output:	*out*	Oktettstring der Länge *n* Oktett
Errors:	–	Keine
Notation:		*out* = Extract_MSByte(*in*, *n*)

Das COS MUSS in den Oktettstring *out* die *n* MSByte von *in* kopieren.**[<=]**

5.8 PaddingIso

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Es wird gemäß [ISO/IEC 7816-4#10.2.3.1] Abschnitt "Sequential stage" Spiegelstrich 2 gepadded. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

A_13589 - (N000.900) K_COS

Tabelle 13: CosT_552: Definition der Funktion PaddingIso(...)

Input:	*in*	Oktettstring mit beliebigem Inhalt und beliebiger Länge
	n	Integer, positive ganze Zahl, gibt die Blocklänge an, auf die *out* zu padden ist
Output:	*out*	Oktettstring mit einer Anzahl Oktette, die Vielfaches von *n* ist
Errors:	–	Keine
Notation:		*out* = PaddingIso(*in*, *n*)

Das COS MUSS folgende Aktion durchführen:
a. Schritt 1: *out* = *in* || '80'.
b. Schritt 2: Wenn 0 == OctetLength(*out*) mod *n*,
dann gebe *out* zurück und beende den Algorithmus,
sonst fahre mit Schritt 3 fort.
c. Schritt 3: *out* = *out* || '00'.
d. Schritt 4: Fahre mit Schritt 2 fort.**[<=]**

Hinweis CosH_cc1: Dies ist die Umkehrfunktion zu (N001.000).

5.9 TruncateIso

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

A_13590 - (N001.000) K_COS

Tabelle 14: CosT_9fb: Definition der Funktion TruncateIso(...)

Input:	*in*	Oktettstring mit beliebigem Inhalt und einer Anzahl Oktette, die Vielfaches von *n* ist
	n	Integer, gibt die Blocklänge in Oktett an, auf die gepadded wurde
Output:	*out*	Oktettstring mit beliebigem Inhalt und beliebiger Länge
Errors:	*paddingError*	Die Länge von *in* ist kein Vielfaches von *n*, oder es ist kein Bit im Oktettstring *in* gesetzt, oder der Oktettstring *in* ist falsch gepadded, oder es sind zu viele Paddingbits vorhanden
Notation:		*out* = TruncateIso(*in*, *n*)

Das COS MUSS folgende Aktion durchführen:
a. Schritt 1: Setze *len* = OctetLength(*in*).
Wenn *len* kein Vielfaches von *n* ist,
dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
b. Schritt 2: Wenn OctetLength(*in*) ~~*len*~~ gleich null ist, dann
breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
c. Schritt 3: Wenn LSByte von *in* den Wert '00' hat,
dann setze *in* = Extract_MSByte(*in*, OctetLength(*in*) – 1)
und fahre mit Schritt 2 fort.
d. Schritt 4: Wenn LSByte von *in* nicht den Wert '80' hat,
dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
e. Schritt 5: *out* = Extract_MSByte(*in*, OctetLength(*in*) – 1)
f. Schritt 6: Wenn (*len* – OctetLength(*out*)) größer als *n* ist,
dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.**[<=]**

Hinweis CosH_523: Dies ist die Umkehrfunktion zu (N000.900).

5.10 MGF Mask Generation Function

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

G2_N001.100 - (N001.100) K_COS

Tabelle 15: CosT_24c: Definition der Funktion MGF(...)

Input:	Z	Bitstring mit beliebigem Inhalt und beliebiger Länge
	L_N	Integer, gibt die Bitlänge von N an
	i	Integer, Startwert der Iteration

Output:	N	Bitstring, Ergebnis der Mask Generation Function
Errors:	*error*	Gemäß [ISO/IEC 9796-2#C.3.2] Punkt 1 ist ein Fehler zu werfen, falls Z zu lang, oder L_N zu groß ist. *Hinweis CosH_ff6: Dieser Fehler ist für Smartcards derzeit nicht praxisrelevant, da er erst dann auftritt, wenn Z oder N oberhalb mehrerer Gigabyte liegen.*
Notation:		N = MGF(Z, L_N , i)

Das COS MUSS bei der Berechnung von N aus Z, L_N und i folgenden Algorithmus ausführen:
a. Schritt 1: Setze n = '', (Anmerkung: Leerer Oktettstring).
b. Schritt 2: Setze n = n || SHA_256(BS2OS(Z) || I2OS(i, 4)).
c. Schritt 3: Wenn die Hash-Operation mit einem Fehler terminierte,
dann breche diesen Algorithmus mit der Fehlermeldung *error* ab.
d. Schritt 4: Setze i = i + 1.
e. Schritt 5: Wenn i größer gleich 2^{32} = '1 0000 0000' ist,
dann breche diesen Algorithmus mit der Fehlermeldung *error* ab.
f. Schritt 6: Wenn das Achtfache von OctetLength(n) kleiner als L_N ist,
dann setze diesen Algorithmus mit Schritt 2 fort.
g. Schritt 7: Setze N = Extract_MSBit(n, L_N).**[<=]**

Hinweis CosH_e55: Wenn diese Funktion mit i gleich 0 aufgerufen wird, dann ist das Ergebnis konform zu
a. [ISO/IEC 9796-2#C], und
b. [PKCS#1#B.2.1].

Hinweis CosH_e90: Wenn diese Funktion mit i gleich 1 aufgerufen wird, dann ist das Ergebnis konform zu
a. [ANSI X9.63#5.6.3], falls dort der optionale Parameter SharedInfo leer ist, und
b. [BSI-TR-03111#4.3.3], allerdings wird dort das Schlüsselmaterial anders aus N extrahiert.

5.11 RAND Zufälliger Oktettstring

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

A_13591 - (N001.200) K_COS

Tabelle 16: CosT_7d4: Definition der Funktion RAND(...)

Input:	n	Positive ganze Zahl, welche die Anzahl Oktette in *out* angibt
Output:	*out*	Zufälliger Oktettstring der Länge n Oktett
Errors:	–	Keine
Notation:		*out* = RAND(n)

Das COS MUSS folgende Aktion durchführen: Erzeuge einen Oktettstring out der Länge n, wobei jedes Bit von out zufällig erzeugt wird.**[<=]**

5.12 ceiling, Aufrunden einer reellen Zahl

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

A_13592 - (N001.210) K_COS

Tabelle 17: CosT_c0e: Definition der Funktion ceiling(...)

Input:	x	beliebige reelle Zahl, negativ, null oder positiv
Output:	n	kleinste ganze Zahl, die nicht kleiner ist als x
Errors:	–	Keine
Notation:		n = ceiling(x)

Das COS MUSS folgende Aktion durchführen: Zu x wird die kleinste ganze Zahl n bestimmt, die nicht kleiner ist als x, das heißt: ceiling(x) = min {n ϵ Z | $n \geq x$}.**[<=]**

5.13 floor, Abrunden einer reellen Zahl

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird die hier beschriebene Funktion wie folgt verwendet:

A_13593 - (N001.220) K_COS

Tabelle 18: CosT_6ed: Definition der Funktion floor(...)

Input:	x	beliebige reelle Zahl, negativ, null oder positiv
Output:	n	größte ganze Zahl, die nicht größer ist als x
Errors:	–	Keine
Notation:		n = floor(x)

Das COS MUSS folgende Aktion durchführen:Zu x wird die größte ganze Zahl n bestimmt, die nicht größer ist als x, das heißt: floor(x) = max {n ϵ Z | $n \leq x$}.**[<=]**

6 Kryptographische Algorithmen (normativ)

6.1 Hash-Algorithmen

Ein Hash-Algorithmus errechnet zu einer beliebigen Folge von Bits, von (beinahe) unbegrenzter Länge, einen Bitstring fester Länge. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird ein Hash-Algorithmus als Funktion, wie in den folgenden Unterkapiteln gezeigt, verwendet.

6.1.1 SHA-1

A_13594 - (N001.290) K_COS

Tabelle 19: CosT_12c: Definition der Funktion SHA_1(...)

Input:	M	Beliebiger Oktettstring, der zu hashen ist
Output:	H	Oktettstring, Hash-Wert der Länge 20 Oktett = 160 Bit
Errors:	"M too long"	Zu hashende Nachricht M enthält zu viele Bits. Dies ist der Fall wenn die Nachricht M länger ist als 2^{64} bit = 2 EiB. *Hinweis CosH_613: Dieser Fehler ist für Smartcards nicht praxisrelevant. Um eine Nachricht mit einer Länge von 2^{64} Bit innerhalb von 10 Jahren zu hashen, wäre der Hash-Algorithmus mit einer Geschwindigkeit von 6,8 GiB pro Sekunde auszuführen.*
Notation:		H = SHA_1(M)

Das COS MUSS H gemäß [FIPS 180-4#6.1] aus M berechnen.**[<=]**

6.1.2 SHA-256

A_13595 - (N001.300) K_COS

Tabelle 20: CosT_159: Definition der Funktion SHA_256(...)

Input:	M	Beliebiger Oktettstring, der zu hashen ist
Output:	H	Oktettstring, Hash-Wert der Länge 32 Oktett = 256 Bit
Errors:	"M too long"	Zu hashende Nachricht M enthält zu viele Bits. Dies ist der Fall wenn die Nachricht M länger ist als 2^{64} bit = 2 EiB *Hinweis CosH_de3: Dieser Fehler ist für Smartcards nicht praxisrelevant. Um eine Nachricht mit einer Länge von 2^{64} Bit innerhalb von 10 Jahren zu hashen, wäre der Hash-Algorithmus mit einer Geschwindigkeit von 6,8 GiB pro Sekunde auszuführen.*
Notation:		H = SHA_256(M)

Das COS MUSS H gemäß [FIPS 180-4#6.2] aus M berechnen.**[<=]**

6.1.3 SHA-384

A_13596 - (N001.310) K_COS

Tabelle 21: CosT_a9f: Definition der Funktion SHA_384(...)

Input:	M	Beliebiger Oktettstring, der zu hashen ist
Output:	H	Oktettstring, Hash-Wert der Länge 48 Oktette = 384 bit
Errors:	"M too long"	Zu hashende Nachricht M enthält zu viele Bits. Dies ist der Fall wenn die Nachricht M länger ist als 2^{128} bit = 2^{125} Byte *Hinweis CosH_f8b: Dieser Fehler ist für Smartcards weniger relevant, als der unter CosH_de3 beschriebene.*
Notation:		H = SHA_384(M)

Das COS MUSS H gemäß [FIPS 180-4#6.5] aus M berechnen.**[<=]**

6.1.4 SHA-512

A_13597 - (N001.320) K_COS

Tabelle 22: CosT_676: Definition der Funktion SHA_512(...)

Input:	M	Beliebiger Oktettstring, der zu hashen ist
Output:	H	Oktettstring, Hash-Wert der Länge 64 Oktette = 512 Bit
Errors:	"M too long"	Zu hashende Nachricht M enthält zu viele Bits. Dies ist der Fall wenn die Nachricht M länger ist als 2^{128} Bit = 2^{125} Byte *Hinweis CosH_7b2: Dieser Fehler ist für Smartcards weniger relevant, als der unter CosH_de3 beschriebene.*
Notation:		H = SHA_512(M)

Das COS MUSS H gemäß [FIPS 180-4#6.4] aus M berechnen.**[<=]**

6.2 Schlüsselvereinbarung

In diesem Abschnitt wird eine Funktion beschrieben, die aus einem Eingabewert Schlüsselmaterial für symmetrische Algorithmen berechnet. Derartiges Schlüsselmaterial wird vorwiegend im Rahmen von sicherer Transportverschlüsselung eingesetzt. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Schlüsselvereinbarung als Funktion, wie in den folgenden Unterkapiteln gezeigt, verwendet.

6.2.1 Verhalten für 3TDES-Schlüssel, Option_DES

(N001.400) ist absichtlich leer.

~~**Tabelle 23, CosT_084: Definition der Funktion KeyDerivation_3TDES(...)**~~

~~Input:~~	~~KD~~	~~Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung. Prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts~~
~~Output:~~	~~K_{enc}~~	~~Oktettstring der Länge 24 Oktette, der als 3TDES-Schlüssel für Ver- und Entschlüsselung verwendet wird~~
	~~T_1~~	~~Nicht-negative ganze Zahl, die im Zusammenhang mit K_{enc} als Send Sequence Counter verwendet wird (siehe CosT_20c und CosT_ef0)~~
	~~K_{mac}~~	~~Oktettstring der Länge 24 Oktette, der als 3TDES-Schlüssel für MAC-Generierung und MAC-Prüfung verwendet wird~~
	~~T_2~~	~~Nicht-negative ganze Zahl, die im Zusammenhang mit K_{mac} als Send Sequence Counter verwendet (siehe (N032.800) und (N034.100))~~

~~Errors:~~	~~–~~	~~Keine~~
~~Notation:~~		~~(K_{enc}, T_1, K_{mac}, T_2) = KeyDerivation_3TDES(KD)~~

A_13598 - (N001.400) K_COS_G1, Option_DES
~~Das COS MUSS (K_{enc}, T_1, K_{mac}, T_2) gemäß [ANSI X9.63#5.6.3] wie folgt aus KD berechnen.~~
~~a. Schritt 1: km = BS2OS(MGF(OS2BS(KD), 512, 1)).~~
~~b. Schritt 2: Teile km so auf K_{enc}, t_1, K_{mac} und t_2 auf, dass gilt:~~
~~1. OctetLength(K_{enc}) == OctetLength(K_{mac}) == 24~~
~~2. OctetLength(t_1) == OctetLength(t_2) == 8~~
~~3. km == K_{enc} || t_1 || K_{mac} || t_2~~
~~c. Schritt 3: T_1 = OS2I(t_1) und T_2 = OS2I(t_2)~~**[<=]**

6.2.2 Vereinbarung von AES-128-Schlüsseln

A_13599 - (N001.500) K_COS

Tabelle 24: CosT_7f9: Definition der Funktion KeyDerivation_AES128(...)

Input:	KD	Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung. Prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts
Output:	K_{enc}	Oktettstring der Länge 16 Oktette, der als AES-128-Schlüssel für Ver- und Entschlüsselung verwendet wird
	K_{mac}	Oktettstring der Länge 16 Oktette, der als AES-128-Schlüssel für MAC-Generierung und MAC-Prüfung verwendet wird
	T_2	Nicht-negative ganze Zahl, die im Zusammenhang mit K_{mac} als Send Sequence Counter $SSCmac$ verwendet wird (siehe (N032.800) und (N034.100))
Errors:	–	keine
Notation:		(K_{enc}, K_{mac}, T_2) = KeyDerivation_AES128(KD)

Das COS MUSS (K_{enc}, K_{mac}, T_2) analog zu [BSI-TR-03110-3#A.2.3.2] wie folgt aus KD berechnen:
a. K_{enc} = Extract_MSByte(SHA_1(KD || '00000001'), 16)
b. K_{mac} = Extract_MSByte(SHA_1(KD || '00000002'), 16)
c. T_2 = 0
[<=]

6.2.3 Vereinbarung von AES-192-Schlüsseln

A_13600 - (N001.510) K_COS

Tabelle 25: CosT_cae: Definition der Funktion KeyDerivation_AES192(...)

Input:	KD	Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung, prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts
Output:	K_{enc}	Oktettstring der Länge 24 Oktette, der als AES-192-Schlüssel für Ver- und Entschlüsselung verwendet wird
	K_{mac}	Oktettstring der Länge 24 Oktette, der als AES-192-Schlüssel für MAC-Generierung und MAC-Prüfung verwendet wird
	T_2	Nicht-negative ganze Zahl, die im Zusammenhang mit K_{mac} als Send Sequence Counter $SSCmac$ verwendet wird (siehe (N032.800) und (N034.100))
Errors:	–	keine
Notation:		(K_{enc}, K_{mac}, T_2) = KeyDerivation_AES192(KD)

Das COS MUSS (K_{enc}, K_{mac}, T_2) analog zu [BSI-TR-03110-3#A.2.3.2] wie folgt aus KD berechnen:
a. K_{enc} = Extract_MSByte(SHA_256 (KD || '00000001'), 24)
b. K_{mac} = Extract_MSByte(SHA_256(KD || '00000002'), 24)
c. T_2 = 0
[<=]

6.2.4 Vereinbarung von AES-256-Schlüsseln

A_13601 - (N001.520) K_COS

Tabelle 26: CosT_a21: Definition der Funktion KeyDerivation_AES256(...)

Input:	KD	Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselvereinbarung, prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts
Output:	K_{enc}	Oktettstring der Länge 32 Oktette, der als AES-256-Schlüssel für Ver- und Entschlüsselung verwendet wird
	K_{mac}	Oktettstring der Länge 32 Oktette, der als AES-256-Schlüssel für MAC-Generierung und MAC-Prüfung verwendet wird
	T_2	Nicht-negative ganze Zahl, die im Zusammenhang mit K_{mac} als Send Sequence Counter $SSCmac$ verwendet wird (siehe (N032.800) und (N034.100))
Errors:	–	keine
Notation:		(K_{enc}, K_{mac}, T_2) = KeyDerivation_AES256(KD)

Das COS MUSS (K_{enc}, K_{mac}, T_2) analog zu [BSI-TR-03110-3#A.2.3.2] wie folgt aus KD berechnen:
a. K_{enc} = SHA_256(KD || '00000001')

b. K_{mac} = SHA_256(KD || '00000002')
c. T_2 = 0**[<=]**

6.2.5 Schlüsselableitung aus einer Card Access Number

A_13602 - (N001.530) K_COS, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD

Tabelle 27: CosT_f06: Definition der Funktion KDF(...)

Input:	KD	Oktettstring, Key Derivation Data, Ausgangsmaterial zur Schlüsselableitung, prinzipiell handelt es sich um einen Oktettstring beliebiger Länge und beliebigen Inhalts
	n	natürliche Zahl, die bei der Schlüsselableitung berücksichtigt wird
	OID	Algorithmuskennung mit einem Wert aus einer Menge in (N102.440)
Output:	K	Oktettstring, je nach OID mit einer Länge von 16 oder 24 oder 32 Oktette, der als AES-Schlüssel verwendet wird
Errors:	–	keine
Notation:		K = KDF(KD, n, OID)

Das COS MUSS K analog zu [BSI-TR-03110-3#A.2.3] wie folgt aus KD, n und OID berechnen: Wenn OID aus der Menge
a. {id-PACE-ECDH-GM-AES-CBC-CMAC-128, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128} ist,
dann gilt: K = Extract_MSByte(SHA_1(KD || I2OS(n, 4)), 16)
b. {id-PACE-ECDH-GM-AES-CBC-CMAC-192, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192} ist,
dann gilt: K = Extract_MSByte(SHA_256(KD || I2OS(n, 4)), 24)
c. {id-PACE-ECDH-GM-AES-CBC-CMAC-256, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256} ist,
dann gilt: K = SHA_256(KD || I2OS(n, 4))**[<=]**

6.3 Symmetrischer Basisalgorithmus für Vertraulichkeit

Ein Verschlüsselungsalgorithmus berechnet zu einem beliebigen Oktettstring (plaintext) mit Hilfe eines geheimen Schlüssels ein Chiffrat (ciphertext). Ein entsprechender Entschlüsselungsalgorithmus berechnet zu einem Chiffrat (ciphertext) mit Hilfe desselben symmetrischen Schlüssels den ursprünglichen Oktettstring (plaintext). In diesem Kapitel wird lediglich die Bearbeitung eines Blocks mittels eines Blockverschlüsselungsalgorithmus behandelt. Die Behandlung von Inputdaten mit beliebiger Länge wird in 6.7 spezifiziert.

6.3.1 Symmetrische Verschlüsselung eines Datenblocks, Option_DES

Die folgenden Punkte sind absichtlich leer: (N001.600), (N001.700), (N001.800), (N001.900).

~~Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Verschlüsselung als Funktion wie folgt verwendet:~~

6.3.1.1 Verschlüsselung mittels DES, Option_DES

~~**Tabelle 28, CosT_c71: Definition der Funktion DES_ENC(...)**~~

~~Input:~~	~~P_f~~	~~Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Klartext~~
	~~K~~	~~Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, der als Schlüssel verwendet wird~~
~~Output:~~	~~C_f~~	~~Oktettstring, verschlüsselte Daten der Länge 8 Oktett~~
~~Errors:~~	~~–~~	~~Keine~~
~~Notation:~~		~~C_f = DES_ENC(K, P_f)~~

A_13603 - (N001.600) K_COS_G1, Option_DES
~~Das COS MUSS C_f mittels K gemäß [ANSI X3.92] aus P_f berechnen.~~**[<=]**

6.3.1.2 Verschlüsselung mittels 3TDES, Option_DES

~~**Tabelle 29, CosT_01b: Definition der Funktion 3TDES_ENC(...)**~~

~~Input:~~	~~P_f~~	~~Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Klartext~~
	~~K~~	~~Beliebiger Oktettstring der Länge 24 Oktett = 192 Bit, der als Schlüssel verwendet wird. K setzt sich zusammen aus den drei Teilschlüsseln Ka, Kb, Kc und es gilt: $K = Ka \ \| \ Kb \ \| \ Kc$~~
~~Output:~~	~~C_f~~	~~Oktettstring, verschlüsselte Daten der Länge 8 Oktett~~
~~Errors:~~	~~–~~	~~Keine~~
~~Notation:~~		~~C_f = 3TDES_ENC(K, P_f)~~

A_13604 - (N001.700) K_COS_G1, Option_DES
~~Das COS MUSS C_f mittels K wie folgt aus P_f berechnen: C_f = DES_ENC(Kc, DES_DEC(Kb, DES_ENC(Ka, P_f))).~~**[<=]**

6.3.2 Symmetrische Entschlüsselung eines Datenblocks, Option_DES

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Entschlüsselung als Funktion wie folgt verwendet:

6.3.2.1 Entschlüsselung mittels DES, Option_DES

Tabelle 30, CosT_a36: Definition der Funktion DES_DEC(...)

Input:	C_i	Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Chiffrat
	K	Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, der als Schlüssel verwendet wird
Output:	P_i	Oktettstring, entschlüsselte Daten der Länge 8 Oktett
Errors:	–	Keine
Notation:		P_i = DES_DEC(K, C_i)

A_13605 - (N001.800) K_COS_G1, Option_DES
Das COS MUSS P_i mittels K gemäß [ANSI X3.92] aus C_i berechnen.[<=]

6.3.2.2 Entschlüsselung mittels 3TDES, Option_DES

Tabelle 31, CosT_478: Definition der Funktion 3TDES_DEC(...)

Input:	C_i	Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Chiffrat
	K	Beliebiger Oktettstring der Länge 24 Oktett = 192 Bit, der als Schlüssel verwendet wird. K setzt sich zusammen aus den drei Teilschlüsseln Ka, Kb, Kc und es gilt: $K = Ka \mid\mid Kb \mid\mid Kc$
Output:	P_i	Oktettstring, entschlüsselte Daten der Länge 8 Oktett
Errors:	–	Keine
Notation:		P_i = 3TDES_DEC(K, C_i)

A_13606 - (N001.900 K_COS_G1, Option_DES
Das COS MUSS P_i mittels K wie folgt aus C_i berechnen: P_i = DES_DEC(Ka, DES_ENC(Kb, DES_DEC(Kc, C_i))).[<=]

6.3.3 Symmetrische Verschlüsselung eines Datenblocks, AES

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Verschlüsselung als Funktion wie folgt verwendet:

A_13607 - (N002.000) K_COS

Tabelle 32: CosT_ee7: Definition der Funktion AES_ENC(...)

Input:	P_j	Beliebiger Oktettstring der Länge 16 Oktett = 128 Bit, Klartext
	K	Beliebiger Oktettstring der Länge 16 oder 24 oder 32 Oktett, der als Schlüssel verwendet wird
Output:	C_j	Oktettstring, verschlüsselte Daten der Länge 16 Oktett
Errors:	–	Keine
Notation:		C_j = AES_ENC(K, P_j)

Das COS MUSS C_j mittels K so aus P_j berechnen, dass sich derselbe funktionale Zusammenhang f_K: P_j -> C_j ergibt wie in [FIPS 197#Figure 5].**[<=]**

6.3.4 Symmetrische Entschlüsselung eines Datenblocks, AES

Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Verschlüsselung als Funktion wie folgt verwendet:

A_13608 - (N002.010) K_COS

Tabelle 33: CosT_bed: Definition der Funktion AES_DEC(...)

Input:	C_j	Oktettstring, verschlüsselte Daten der Länge 16 Oktett
	K	Beliebiger Oktettstring der Länge 16 oder 24 oder 32 Oktett, der als Schlüssel verwendet wird
Output:	P_j	Beliebiger Oktettstring der Länge 16 Oktett = 128 Bit, Klartext
Errors:	–	Keine
Notation:		P_j = AES_DEC(K, C_j)

Das COS MUSS P_j mittels K so aus C_j berechnen, dass sich derselbe funktionale Zusammenhang f_K: C_j → P_j ergibt, wie in [FIPS 197#Figure 12] oder [FIPS 197#Figure 15].**[<=]**

6.4 Asymmetrischer Basisalgorithmus RSA

RSA ist ein im COS verwendeter Basisalgorithmus. Bezüglich des mathematischen Hintergrundes wird an dieser Stelle lediglich auf [PKCS#1] verwiesen.

G2_N002.100 - (N002.100) K_COS
Das COS MUSS mindestens RSA–Schlüssel mit
a. einer Moduluslänge $modulusLength$ aus den Mengen
   1. {2048} Bit für private und öffentliche RSA-Schlüssel unterstützen und
   2. {3072} Bit, falls es sich um einen privaten RSA-Schlüssel handelt und
b. öffentlichen Exponenten e aus folgendem Intervall unterstützen:
   $[2^{16}+1, 2^{32}-1]$ = [65.537, 4.294.967.295] = ['0001 0001', 'FFFF FFFF']. **[<=]**

(N002.200) Diese Anforderung ist absichtlich leer. Der ursprüngliche Text erlaubte auch andere Moduluslängen zu unterstützen. Dies wird nun in (N002.100) durch das Wort "mindestens" ausgedrückt.

(N002.300) Diese Anforderung ist absichtlich leer. Der ursprüngliche Text erlaubte auch andere Werte für öffentliche Exponenten. Dies wird nun in (N002.100) durch das Wort "mindestens" ausgedrückt.

A_13609 - (N002.400) COS
~~Private RSA-Schlüssel, welche die in [BKryA#3.1] empfohlenen Schranken ε 1 und ε 2 verletzen, für die mithin NICHT gilt 0,1 < |log 2 (p) – log 2 (q)| < 30,~~
~~a. KÖNNEN vom COS akzeptiert werden.~~
~~b. KÖNNEN vom COS abgelehnt werden.~~
Das COS MUSS für die beiden Primfaktoren p und q des Modulus Intervalle unterstützen, die für die Zahl
epsilon = $log_2(q) - log_2(p)$ mindestens Werte aus dem Intervall [1/10, 1/2] umfassen. **[<=]**

Hinweis CosH_e24: Gemäß (N002.400) sind andere, beliebige Verhältnisse von p und q funktional zulässig. Insbesondere auch solche, die zu $p > q$ gehören. Zudem handelt es sich mitnichten um eine funktionale Vorgabe für p und q im Rahmen der Onboard-Schlüsselgenerierung. Die Anforderung (N002.400) hat lediglich die Aufgabe herstellerübergreifend einen Bereich für p und q festzulegen, der im Rahmen einer externen Schlüsselgenerierung mit dem COS interoperabel ist (etwa für Testlaborkartenspezifikation, Personalisierung und ähnliches.

Konform zu [PKCS#1] Kapitel 2 werden die RSA-Schlüsselparameter in diesem Dokument wie folgt dargestellt:

Tabelle 34: CosT_689: Liste der Schlüsselparameter eines RSA-Schlüssels

Parameter	Bedeutung
n	RSA Modulus
e	RSA öffentlicher Exponent
d	RSA privater Exponent

Hinweis CosH_e02: In diesem Dokument wird der Einfachheit halber nicht mit "chinese remainder theorem" Parametern gearbeitet. Wegen der größeren Performanz wird aber empfohlen, innerhalb des COS für Operationen mit dem privaten Schlüssel "chinese remainder theorem" Parameter zu verwenden.

6.5 Asymmetrischer Basisalgorithmus elliptische Kurven

Kryptographie basierend auf elliptischen Kurven ist ein im COS verwendeter Basisalgorithmus. Bezüglich des mathematischen Hintergrundes wird an dieser Stelle lediglich auf [BSI-TR-03111] verwiesen. Die Domainparameter der im folgenden genannten Kurven finden sich auch in Kapitel 19.

G2_N002.500.a.1 - (N002.500)a.1 K_COS
Das COS MUSS die folgende elliptische Kurve unterstützen: [RFC5639#3.4, brainpoolP256r1], (siehe CosT_de0).**[<=]**

G2_N002.500.a.2 - (N002.500)a.2 K_COS
Das COS MUSS die folgende elliptische Kurve unterstützen: [ANSI X9.62#L.6.4.3, ansix9p256r1], (siehe CosT_f68).**[<=]**

G2_N002.500.b.1 - (N002.500)b.1 K_COS
Das COS MUSS die folgende elliptische Kurve unterstützen: [RFC5639#3.6, brainpoolP384r1], (siehe CosT_62f).**[<=]**

G2_N002.500.b.2 - (N002.500)b.2 K_COS
Das COS MUSS die folgende elliptische Kurve unterstützen: [ANSI X9.62#L6.5.2, ansix9p384r1], (siehe CosT_88a).**[<=]**

G2_N002.500.c - (N002.500)c K_COS
Das COS MUSS die folgende elliptische Kurve unterstützen: [RFC5639#3.7, brainpoolP512r1], (siehe CosT_198).**[<=]**

A_13610 - (N002.609) K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere elliptische Kurven
a. unterstützt oder
b. ablehnt.**[<=]**

6.6 Datenauthentisierung

Im Rahmen einer Datenauthentisierung werden beliebigen Daten weitere Informationen derart hinzugefügt, dass die Integrität und Authentizität der Daten überprüfbar ist.

6.6.1 MAC-Generierung

Die MAC-Generierung ist eine Datenauthentisierung, welche auf einem symmetrischen Basisalgorithmus basiert. Dabei wird zu einem Oktettstring beliebigen Inhalts und Länge ein MAC berechnet, dessen Länge lediglich vom Algorithmus abhängt, nicht aber vom Oktettstring. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine MAC-Generierung als Funktion wie folgt verwendet:

6.6.1.1 Generierung Retail-MAC, Option_DES

(N002.700) ist absichtlich leer.

~~**Tabelle 35, CosT_d3d: Definition der Funktion CALCULATE_Retail_MAC(...)**~~

~~Input:~~	~~M~~	~~Beliebiger Oktettstring beliebiger Länge für den ein MAC berechnet wird~~
	~~K~~	~~Beliebiger Oktettstring der Länge 24 Oktett = 192 Bit, der als Schlüssel verwendet wird. K setzt sich zusammen aus den drei Teilschlüsseln Ka, Kb, Kc und es gilt: $K = Ka \parallel Kb \parallel Kc$~~
~~Output:~~	~~T~~	~~Oktettstring, der zur Prüfung der Integrität und Authentizität von M verwendbar ist~~
~~Errors:~~	~~–~~	~~Keine~~
~~Notation:~~		~~T = CALCULATE_Retail_MAC(K, M)~~

A_13611 - (N002.700) K_COS_G1, Option_DES

~~Das COS MUSS T mittels K aus M berechnen. Dabei werden folgende Schritte durchgeführt:~~

~~a. Schritt 1: Berechne X = PaddingIso(M, 8)~~

~~b. Schritt 2: Teile X auf in Blöcke mit jeweils 8 Oktett $X = X_1 \parallel X_2 \parallel \ldots \parallel X_n$.~~

~~c. Schritt 3: Setze Y_0 = '0000 0000 0000 0000'.~~

~~d. Schritt 4: Berechne Y_i = DES_ENC(Ka, Y_{i-1} XOR X_i) für i = 1, …, n – 1.~~

~~e. Schritt 5: Berechne T = 3TDES_ENC(K, Y_{n-1} XOR X_n).~~**[<=]**

~~*Hinweis CosH_dc6: Im Rahmen von Secure Messaging wird ein von "null" verschiedener Initialisierungsvektor Y_0 bei der Konstruktion der Nachricht M berücksichtigt (siehe dazu (N032.800) und (N034.100)).*~~

6.6.1.2 Generierung CMAC ohne internes Padding vor der CMAC-Berechnung

A_13612 - (N002.810) K_COS

Tabelle 36: CosT_fd0: Definition der Funktion CalculateCMAC_NoPadding(...)

Input:	M	Beliebiger Oktettstring, für den ein MAC berechnet wird
	K	Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von K beträgt 16 oder 24 oder 32 Oktett
Output:	T	Oktettstring, der zur Prüfung der Integrität und Authentizität von M verwendbar ist
Errors:	–	Keine
Notation:		T = CalculateCMAC_NoPadding(K, M)

Das COS MUSS T mittels K gemäß [CMAC#6.2] aus M berechnen. Dabei werden folgende Schritte durchgeführt:

a. Definitionen: CIPH = Block Cipher Algorithmus (siehe (N002.000))
b = Blocklänge von CIPH in Bit hier gleich 128 Bit
M_{len} = OctetLength(M).

b. Schritt 1: Generiere Unterschlüssel $K1$ und $K2$ gemäß [CMAC#6.1].

c. Schritt 2: Wenn M_{len} gleich null ist,
dann setze n = 1,
sonst r = M_{len} mod b und
wenn r == 0,
dann setze n = M_{len} / b
sonst setze n = (M_{len} – r) / b + 1.
d. Schritt 3: Teile M wie folgt in Blöcke auf: M == M_1 || M_2 || … || M_{n-1} || M^*_n.
Die Blöcke M_1 bis M_{n-1} besitzen die Länge b. Der Block M^*_n besitzt eine Länge kleiner gleich b.
e. Schritt 4: Wenn r == 0 ist,
dann setze M_n = $K1$ XOR M^*_n,
sonst setze M_n = $K2$ XOR PaddingIso(M^*_n, b / 8).
f. Schritt 5: C_0 = I2OS(0, b / 8)
g. Schritt 6: C_i = AES_ENC(K, C_{i-1} XOR M_i).
h. Schritt 7: T = Extract_MSByte(C_n, 8).**[<=]**

6.6.1.3 Generierung CMAC mit internem ISO-Padding vor der CMAC-Berechnung A_13613 - (N002.820) K_COS

Tabelle 37: CosT_245: Definition der Funktion CalculateCMAC_IsoPadding(...)

Input:	M	Beliebiger Oktettstring, für den ein MAC berechnet wird
	K	Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von K beträgt 16 oder 24 oder 32 Oktett.
Output:	T	Oktettstring, der zur Prüfung der Integrität und Authentizität von M verwendbar ist
Errors:	–	Keine
Notation:		T = CalculateCMAC_IsoPadding(K, M)

Das COS MUSS T mittels K gemäß [CMAC#6.2] aus M wie folgt berechnen.
T = CalculateCMAC_NoPadding(K, PaddingIso(M, 16)).**[<=]**

6.6.2 MAC-Prüfung

Die MAC-Prüfung prüft die Konsistenz zwischen Daten und den hinzugefügten Informationen der Datenauthentisierung. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine MAC-Prüfung als Funktion wie folgt verwendet:

6.6.2.1 Prüfung Retail-MAC, Option_DES

(N002.900) ist absichtlich leer.

~~**Tabelle 38, CosT_590: Definition der Funktion VERIFY_Retail_MAC(...)**~~

~~Input:~~	~~*M*~~	~~Beliebiger Oktettstring, der durch einen MAC geschützt ist~~
	~~*T'*~~	~~Oktettstring, der *M* zwecks Datenauthentisierung hinzugefügt wurde~~
	~~*K*~~	~~Beliebiger Oktettstring der Länge 24 Oktett = 192 Bit, der als Schlüssel verwendet wird. *K* setzt sich zusammen aus den drei Teilschlüsseln *Ka*, *Kb*, *Kc* und es gilt: *K = Ka || Kb || Kc*~~
~~Output:~~	~~*out*~~	~~Ergebnis der MAC-Prüfung, entweder *VALID* oder *INVALID*~~
~~Errors:~~	~~–~~	~~Keine~~
~~Notation:~~		~~*out* = VERIFY_Retail_MAC(*K*, *T'*, *M*)~~

A_13614 - (N002.900) K_COS_G1, Option_DES

~~Das COS MUSS die Integrität der Daten *M* mittels *T'* und *K* prüfen. Dabei werden folgende Schritte durchgeführt:~~
~~a. Schritt 1: *T* = CALCULATE_Retail_MAC(*K*, *M*).~~
~~b. Schritt 2: Wenn *T* identisch zu *T'* ist, dann gebe *VALID* zurück sonst gebe *INVALID* zurück.~~**[<=]**

6.6.2.2 Prüfung CMAC ohne internes Padding vor der CMAC-Berechnung

A_13615 - (N003.010) K_COS

Tabelle 39: CosT_fa6: Definition der Funktion VerifyCMAC_NoPadding(...)

Input:	M	Beliebiger Oktettstring, der durch einen MAC geschützt ist
	T'	Oktettstring, der M zwecks Datenauthentisierung hinzugefügt wurde
	K	Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von K beträgt 16 oder 24 oder 32 Oktett
Output:	*out*	Ergebnis der MAC-Prüfung, entweder *VALID* oder *INVALID*
Errors:	–	Keine
Notation:		*out* = VerifyCMAC_NoPadding(K, T', M)

Das COS MUSS die Integrität der Daten M mittels T' und K gemäß [CMAC#6.3] prüfen. Dabei werden folgende Schritte durchgeführt:
a. Schritt 1: T = CalculateCMAC_NoPadding(K, M).
b. Schritt 2: Wenn T identisch zu T' ist, dann gebe *VALID* zurück sonst gebe *INVALID* zurück.**[<=]**

6.6.2.3 Prüfung CMAC mit internem Padding vor der CMAC Berechnung

A_13616 - (N003.020) K_COS

Tabelle 40: CosT_2ad: Definition der Funktion VerifyCMAC_IsoPadding(...)

Input:	M	Beliebiger Oktettstring, der durch einen MAC geschützt ist
	T'	Oktettstring, der M zwecks Datenauthentisierung hinzugefügt wurde
	K	Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von K beträgt 16 oder 24 oder 32 Oktett
Output:	out	Ergebnis der MAC-Prüfung, entweder $VALID$ oder $INVALID$
Errors:	–	Keine
Notation:		out = VerifyCMAC_IsoPadding(K, T', M)

Das COS MUSS die Integrität der Daten M mittels T' und K gemäß [CMAC#6.3] prüfen. Dabei werden folgende Schritte durchgeführt:
a. Schritt 1: T = CalculateCMAC_IsoPadding(K, M).
b. Schritt 2: Wenn T identisch zu T' ist, dann gebe $VALID$ zurück sonst gebe $INVALID$ zurück.**[<=]**

6.6.3 Signaturberechnung

Unter der Berechnung einer Signatur wird in diesem Dokument lediglich die mit dem privaten Schlüssel durchgeführte Operation verstanden.

6.6.3.1 Signaturberechnung mittels RSA

6.6.3.1.1 RSA, ISO9796–2, DS1, SIGN, Option_RSA_CVC

(N003.100) ist absichtlich leer.

~~Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos INTERNAL AUTHENTICATE sichtbar (siehe (N086.900)c und (N086.900)d). Sie wird wie folgt verwendet:~~

~~**Tabelle 41, CosT_533: Definition der Funktion RSA_ISO9796_2_DS1_SIGN**~~

~~Input:~~	~~PrK~~	~~Privater RSA-Schlüssel gemäß (N008.800)~~
	~~M~~	~~Beliebiger Oktettstring, der die zu signierende Nachricht repräsentiert~~
~~Output:~~	~~sig~~	~~Oktettstring, welcher die Signatur repräsentiert~~
	~~M_2~~	~~Oktettstring, welcher den "non recoverable part" der Nachricht M repräsentiert~~

~~Errors:~~	~~–~~	~~–~~
~~Notation:~~		~~(sig, M_2) = RSA_ISO9796_2_DS1_SIGN(PrK, M)~~

A_13617 - (N003.100) K_COS_G1, Option_RSA_CVC

~~Das COS MUSS folgende Aktionen gemäß [ISO/IEC 9796-2#7, 8] durchführen, wobei folgende Definitionen gelten:~~

~~$n = PrK.n$, $d = PrK.d$~~

~~a. Schritt 1: Message allocation:~~

~~Aufspalten der Nachricht M in M_1 und M_2 gemäß [ISO/IEC 9796-2#7.2.2].~~

~~b. Schritt 2: Message representative production:~~

~~Berechnen des Repräsentanten F der Nachricht M gemäß [ISO/IEC 9796-2#8.3],~~

~~wobei t = 1 gesetzt wird und als Hash-Funktion SHA-256 (siehe (N001.300)) verwendet wird.~~

~~c. Schritt 3: Signature production:~~

~~Gemäß [ISO/IEC 9796-2#7.2.4] und [ISO/IEC 9796-2#B.4] werden folgende Operationen ausgeführt:~~

~~1. Schritt 3.1: $J = OS2I(F)$~~
~~2. Schritt 3.2: $a = J^d \bmod n$~~
~~3. Schritt 3.3: $b = n - a$~~
~~4. Schritt 3.4: $c = \min\{a, b\}$~~
~~5. Schritt 3.5: $sig = I2OS(c, OctetLength(n))$~~**[<=]**

6.6.3.1.2 RSA, SSA, PKCS1–V1_5

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Compute Digital Signature sichtbar (siehe (N088.600)d). Sie wird wie folgt verwendet:

A_13618 - (N003.200) K_COS

Tabelle 42: CosT_234: Definition der Funktion RSASSA_PKCS1_V1_5_SIGN(...)

Input:	*digestInfo*	Beliebiger Oktettstring, der als Digest Info verwendet wird (siehe [PKCS#1] Anhang A.2.4)
	PrK	Privater RSA-Schlüssel gemäß (N008.800)
Output:	*S*	Oktettstring, welcher die Signatur repräsentiert
Errors:	DigestInfoTooLong	Der Inputparameter *digestInfo* enthält zu viele Oktette
Notation:		*S* = RSASSA_PKCS1_V1_5_SIGN(*PrK*, *digestInfo*)

Das COS MUSS folgende Aktion gemäß [PKCS#1] Kapitel 8.2.1 und 9.2 durchführen, wobei folgende Definitionen gelten: $n = PrK.n$, $d = PrK.d$

a. Schritt 0: Wenn OctetLength(*digestInfo*) > 0,4 OctetLength(n) ist,
dann breche diesen Algorithmus mit der Fehlermeldung DigestInfoTooLong ab.

b. Schritt 1: Setze $EM \leftarrow$ '00' || *digestInfo*

c. Schritt 2: Setze $EM \leftarrow$ 'FF' || EM

d. Schritt 3: Wenn OctetLength(EM) kleiner als OctetLength(n) – 2 ist, dann fahre mit Schritt 2 fort.
e. Schritt 4: Setze EM ← '01' || EM
f. Schritt 5: Setze m ← OS2I(EM)
g. Schritt 6: Setze s ← m^d mod n
h. Schritt 7: Setze S ← I2OS(s, OctetLength(n)).**[<=]**

6.6.3.1.3 RSA, SSA, PSS

Diese Funktionalität ist an der physikalischen Schnittstelle nicht sichtbar. Sie wird im Rahmen interner Funktionen verwendet.

G2_N003.300 - (N003.300) K_COS

Tabelle 43: CosT_858: Definition der Funktion RSA_PSS_SIGN(...)

Input:	M_1	Beliebiger Oktettstring, der den "recoverable part" der zu signierenden Nachricht$M$ repräsentiert
	h(M_2)	Beliebiger Oktettstring, der den Hash-Wert über den "non recoverable part" der zu signierenden Nachricht M enthält
	PrK	Privater RSA-Schlüssel gemäß (N008.800)
Output:	sig	Oktettstring, welcher die Signatur repräsentiert
Errors:	–	Keine
Notation:		sig = RSA_PSS_SIGN(PrK, M_1, h(M_2))

Das COS MUSS folgende Aktionen gemäß [ISO/IEC 9796-2#7, 9] durchführen, wobei folgende Definitionen gelten: $n = PrK.n$, $d = PrK.d$
a. Schritt 1: Message representative production:
Berechnen des Repräsentanten F der Nachricht M gemäß [ISO/IEC 9796-2#9.3], wobei
t = 1 und $L_s = L_h$ gesetzt wird und als Hash-Funktion SHA-256 verwendet wird.
Im Einzelnen:
1. Setze C = I2OS(BitLength(OS2BS(M_1)), 8)
2. Setze S = RAND(32)
3. Berechne H = SHA_256(C || M_1 || h(M_2) || S)
4. Berechne F gemäß [ISO/IEC 9796-2#9.3.2].

b. Schritt 2: Signature production: Gemäß [ISO/IEC 9796-2#7.2.4] und [ISO/IEC 9796-2#B.6] werden folgende Operationen ausgeführt:
1. Schritt 2.1: J = OS2I(F)
2. Schritt 2.2: a = J^d mod n
3. Schritt 2.3: sig = I2OS(a, OctetLength(n))**[<=]**

6.6.3.1.4 RSA, ISO9796–2, DS2, SIGN

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Compute Digital Signature sichtbar (siehe (N088.600)b). Sie wird wie folgt verwendet:

A_13619 - (N003.400) K_COS

Tabelle 44: CosT_ace: Definition der Funktion RSA_ISO9796_2_DS2_SIGN(...)

Input:	M_1	Beliebiger Oktettstring, der den "recoverable part" der zu signierenden Nachricht M repräsentiert
	h(M_2)	Beliebiger Oktettstring, der den Hash-Wert über den "non recoverable part" der zu signierenden Nachricht$M$ enthält
	PrK	Privater RSA-Schlüssel gemäß (N008.800)
Output:	sig	Oktettstring, welcher die Signatur repräsentiert
Errors:	–	Keine
Notation:		sig = RSA_ISO9796_2_DS2_SIGN(PrK, M_1, h(M_2))

Das COS MUSS folgende Aktionen gemäß [ISO/IEC 9796-2#7, 9] durchführen, wobei folgende Definition gilt: $n = PrK.n$
a. Schritt 1: a = OS2I(RSA_PSS_SIGN(PrK, M_1, h(M_2))).
b. Schritt 2: b = $n - a$
c. Schritt 3: c = min{ a, b }
d. Schritt 4: sig = I2OS(c, OctetLength(n))**[<=]**

6.6.3.1.5 RSASSA–PSS–SIGN

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen der Kommandos INTERNAL AUTHENTICATE (siehe (N086.900)a) und PSO Compute Digital Signature sichtbar (siehe (N088.600)c). Sie wird wie folgt verwendet:

A_13620 - (N003.500) K_COS

Tabelle 45: CosT_7cc: Definition der Funktion RSASSA_PSS_SIGN(...)

Input:	$mHash$	Beliebiger Oktettstring, der den Hash-Wert über die zu signierenden Nachricht M repräsentiert
	PrK	Privater RSA-Schlüssel gemäß (N008.800)
Output:	S	Oktettstring, welcher die Signatur repräsentiert
Errors:	–	Keine
Notation:		S = RSASSA_PSS_SIGN(PrK, $mHash$)

Das COS MUSS folgende Aktion gemäß [PKCS#1#8.1.1, 9.1.1] durchführen:
a. Schritt 1: M_1 = '', (Anmerkung:M_1 ist ein leerer Oktettstring)
b. Schritt 2: S = RSA_PSS_SIGN(PrK, M_1, $mHash$).**[<=]**

6.6.3.2 Signaturberechnung mittels ELC

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Compute Digital Signature sichtbar (siehe (N088.600)c). Sie wird wie folgt verwendet:

A_13621 - (N003.600) K_COS

Tabelle 46: CosT_78b: Definition der Funktion ELC_SIG(...)

Input:	H	Oktettstring mit beliebigem Inhalt, der einen Hash-Wert repräsentiert
	PrK	Privater ELC–Schlüssel gemäß (N008.900) und (N009.000)
Output:	R	Oktettstring, erster Teil der ECDSA-Signatur
	S	Oktettstring, zweiter Teil der ECDSA-Signatur
Errors:	–	Keine
Notation:		(R, S) = ELC_SIG(PrK, H)

Das COS MUSS folgende Aktionen gemäß [BSI-TR-03111#4.2.1.1] durchführen, wobei folgende Definitionen gelten:
$d_A = PrK.d$, $G = PrK.domainParameter.G$, $n = PrK.domainParameter.n$, $t = PrK.domainParameter.t$
a. Schritt 0: H_t = BS2OS(Extract_MSBit(H, t)).
b. Schritt 1: k = RNG({1,2, …, n – 1}), d.h. zufällig erzeugte ganze Zahl aus dem Intervall [1, n – 1].
c. Schritt 2: Q = [k] G mit $Q = (x_Q, y_Q)$.
d. Schritt 3: r = OS2I(FE2OS(x_Q)) mod n. Wenn r gleich null ist, dann gehe zu Schritt 1.
e. Schritt 4: $k_{inv} = k^{-1}$ mod n.
f. Schritt 5: s = k_{inv} ($r\ d_A$ + OS2I(H_t)) mod n. Wenn s gleich null ist, dann gehe zu Schritt 1.
g. Schritt 6: R = I2OS(r, ceiling(t / 8)).
S = I2OS(s, ceiling(t / 8)).**[<=]**

Hinweis CosH_b8f: In (N003.600)a wird ein möglicherweise zu langer Hash-Wert auf die Bitlänge von n reduziert, vergleiche dazu auch [BSI-TR-03111#4.2]. Bei den in diesem Dokument verwendeten Kombinationen aus Hash-Algorithmus und Kurvenlänge ist H_t stets identisch zu H, sodass die Berechnung in (N003.600)a nicht praxisrelevant ist.

6.6.4 Signaturprüfung

Unter der Prüfung einer Signatur wird in diesem Dokument lediglich die mit dem öffentlichen Schlüssel durchgeführte Operation verstanden. Diese Funktionalität wird an der physikalischen Schnittstelle nur teilweise sichtbar, je nach Schlüsseltyp. Im Rahmen diverser interner Operationen im Betriebssystem wird eine Signaturprüfung als Funktion wie folgt verwendet:

6.6.4.1 RSA, ISO9796–2, DS1, VERIFY, Option_RSA_CVC

Die folgenden Punkte sind absichtlich leer: (N003.700).

~~**Tabelle 47, CosT_adc: Definition der Funktion RSA_ISO9796_2_DS1_VERIFY(...)**~~

~~Input:~~	~~PuK~~	~~Öffentlicher RSA-Schlüssel gemäß (N009.100), (N00.9110) und (N009.200)~~
	~~sig~~	~~Beliebiger Oktettstring, der eine Signatur repräsentiert~~

	M_2	Beliebiger Oktettstring, der den "non recoverable part" der Nachricht$M$ repräsentiert. Es ist möglich, dass M_2 leer ist.
Output:	*out*	Boolean, *True*, falls die Signatur gültig ist, andernfalls *False*
	M	Oktettstring, die rekonstruierte Nachricht
Errors:	–	Keine
Notation:		(*out*, M) = RSA_ISO9796_2_DS1_VERIFY(PuK, *sig*, M_2)

A_13622 - (N003.700) K_COS, Option_RSA_CVC
Das COS MUSS folgende Aktionen gemäß [ISO/IEC 9796-2#7.2] durchführen, wobei folgende Definitionen gelten:
$n = PuK.n$, $e = PuK.e$.**[<=]**

(N003.700)a. Schritt 1: Signature opening: Gemäß [ISO/IEC 9796-2#B.5]:

G2_N003.700.a.1 - (N003.700)a.1 K_COS, Option_RSA_CVC
Schritt 1.1: Wenn die Bitlänge von *sig* ungleich der Bitlänge von n ist, dann MUSS das COS *out* = *False* zurückgeben und den Algorithmus abbrechen.**[<=]**

G2_N003.700.a.2 - (N003.700)a.2 K_COS, Option_RSA_CVC
Schritt 1.2: Wenn das höchstwertige Bit von *sig* den Wert 1 hat, dann MUSS das COS *out* = *False* zurückgeben und den Algorithmus abbrechen.**[<=]**

G2_N003.700.a.3 - (N003.700)a.3 K_COS, Option_RSA_CVC
Schritt 1.3: Das COS MUSS s aus *sig* wie folgt berechnen: s = OS2I(*sig*)**[<=]**

G2_N003.700.a.4 - (N003.700)a.4 K_COS, Option_RSA_CVC
Schritt 1.4: Das COS MUSS J^* aus s wie folgt berechnen: $J^* = s^e \bmod n$.**[<=]**

G2_N003.700.a.5 - (N003.700)a.5 K_COS, Option_RSA_CVC
Schritt 1.5: Das COS MUSS I^* aus J^* wie folgt berechnen: Wenn J^* gerade ist, dann setze $I^* = J^*$, sonst setze $I^* = n - J^*$.**[<=]**

G2_N003.700.a.6 - (N003.700)a.6 K_COS, Option_RSA_CVC
Schritt 1.6: Wenn I^* mod 256 ungleich 'BC' ist, dann MUSS das COS*out* = *False* und M = '' zurückgeben und den Algorithmus abbrechen.**[<=]**

G2_N003.700.a.7 - (N003.700)a.7 K_COS, Option_RSA_CVC
Schritt 1.7: Das COS MUSS F^* aus I^* wie folgt berechnen: F^* = I2OS(I^*, OctetLength(n)).**[<=]**

G2_N003.700.a.8 - (N003.700)a.8 K_COS, Option_RSA_CVC
Schritt 1.8: Wenn das höchstwertige Bit von F^* den Wert 1 hat, dann MUSS das COS *out* = *False* und M = '' zurückgeben und den Algorithmus abbrechen.**[<=]**

(N003.700)b. Schritt 2: Message recovery: Gemäß [ISO/IEC 9796-2#8.4]:

G2_N003.700.b.1 - (N003.700)b.1 K_COS, Option_RSA_CVC
Schritt 2.1: Wenn das zweithöchste Bit von F^* den Wert 0 hat,dann MUSS das COS *out* = *False* und M = '' zurückgeben und den Algorithmus abbrechen.**[<=]**

G2_N003.700.b.2 - (N003.700)b.2 K_COS, Option_RSA_CVC
~~Schritt 2.2: Das COS MUSS H^* und M_1^* gemäß [ISO/IEC 9796-2#8.4] berechnen.~~**[<=]**

G2_N003.700.b.3 - (N003.700)b.3 K_COS, Option_RSA_CVC
~~Schritt 2.3: Das COS MUSS m aus M_1^* und M_2 wie folgt berechnen: $m = M_1^* \| M_2$.~~**[<=]**

G2_N003.700.b.4 - (N003.700)b.4 K_COS, Option_RSA_CVC
~~Schritt 2.4: Das COS MUSS H aus m wie folgt berechen: H = SHA_256(m).~~**[<=]**

G2_N003.700.b.5 - (N003.700)b.5 K_COS, Option_RSA_CVC
~~Schritt 2.5: Das COS MUSS out und M wie folgt berechnen und dann zurückgeben: Wenn H identisch ist zu H^*, dann gilt $out = True$ und $M = m$, sonst gilt $out = False$ und $M =$ ".~~**[<=]**

~~*Hinweis CosH_6a8: ACHTUNG: In (N003.700)b.i ist F^* ein Oktettstring. Aus diesem F^* geht durch Abschneiden des "most significant bit" ein Bitstring hervor, derF^* aus [ISO/IEC 9796-2#8.4] entspricht. Deshalb wird in (N003.700)b.i das zweithöchste Bit geprüft und in [ISO/IEC 9796-2#8.4] das "leftmost" Bit.*~~

6.6.4.2 Signaturprüfung mittels elliptischer Kurven

A_13623 - (N003.800) K_COS

Tabelle 48: CosT_b2b: Definition der Funktion ELC_VER_SIG(...)

Input:	PuK	Öffentlicher ELC-Schlüssel gemäß (N009.300) und (N009.400)
	H	Beliebiger Oktettstring, der einen Hash-Wert repräsentiert
	R	Oktettstring, erster Teil der ECDSA-Signatur
	S	Oktettstring, zweiter Teil der ECDSA-Signatur
Output:	out	Boolean, $True$, falls die Signatur gültig ist, andernfalls $False$
Errors:	–	Keine
Notation:		out = ELC_VER_SIG(PuK, R, S, H)

Das COS MUSS folgende Aktionen gemäß [BSI-TR-03111#4.2.1.2] durchführen, wobei folgende Definitionen gelten:
P_a = $PuK.P$, G = $PuK.domainParameter.G$, n = $PuK.domainParameter.n$, t = $PuK.domainParameter.t$ **[<=]**

A_13624 - (N003.800)a K_COS
Schritt 0: H_t = BS2OS(Extract_MSBit(H, t)), siehe CosH_b8f
r = OS2I(R).
s = OS2I(S).**[<=]**

G2_N003.800.b - (N003.800)b K_COS
Schritt 1: Wenn r oder s nicht Element der Menge {1, 2, …, n – 1} sind, dann gibt die Funktion out = $False$ zurück und bricht diesen Algorithmus ab.**[<=]**

A_17566 - (N003.800)c K_COS
Schritt 2: s_{inv} = s^{-1} mod n. **[<=]**

A_17567 - (N003.800)d K_COS

Schritt 3: u_1 = s_{inv} OS2I (H_t) mod n.
u_2 = s_{inv} r mod n. **[<=]**

A_17568 - (N003.800)e K_COS
Schritt 4: Q = [u_1] G + [u_2] P_a mit Q = (x_Q, y_Q).
Wenn Q gleich dem unendlich fernen Punkt O entspricht,
dann gibt die Funktion out = $False$ zurück und bricht diesen Algorithmus ab.**[<=]**

A_17569 - (N003.800)f K_COS
Schritt 5: v = OS2I (FE2OS(x_Q)) mod n. **[<=]**

G2_N003.800.g - (N003.800)g K_COS
Schritt 6: Das COS MUSS als Ergebnis der Funktion out = $True$ zurückgeben, falls v gleich r ist, andernfalls out = $False$.**[<=]**

6.7 Vertraulichkeit von Daten, symmetrischer Fall

6.7.1 Symmetrische Verschlüsselung

Die symmetrische Verschlüsselung überführt eine beliebige Nachricht *plaintext* (Oktettstring beliebigen Inhalts und Länge) in ein Chiffrat *ciphertext*. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine symmetrische Verschlüsselung als Funktion wie folgt verwendet:

6.7.1.1 Verschlüsselung 3TDES, Option_DES

(N003.900) ist absichtlich leer.

~~**Tabelle 49, CosT_20c: Definition der Funktion 3TDES_CBX_ENC(...)**~~

~~Input:~~	~~P~~	~~Oktettstring, Klartext (*plaintext*), beliebiger Oktettstring beliebiger Länge, der verschlüsselt wird~~
	~~K~~	~~Beliebiger Oktettstring der Länge 24 Oktett, der als Schlüssel verwendet wird~~
	~~T_I~~	~~Beliebige nicht-negative Zahl, die als Startwert verwendet wird~~
~~Output:~~	~~C~~	~~Oktettstring, Chiffrat (*ciphertext*)~~
~~Errors:~~	~~lengthError~~	~~Die Länge von P ist kein ganzzahliges Vielfaches der Blocklänge~~
~~Notation:~~		~~C = 3TDES_CBC_ENC(K, T_I, P)~~

A_13625 - (N003.900) K_COS_G1, Option_DES
~~Das COS MUSS C mittels K und T_I wie folgt aus P berechnen. Dabei sind folgende Aktionen durchzuführen:~~

~~a. Schritt 1: Wenn OctetLength(P) mod 8 ungleich 0 ist,~~
~~dann gebe den Fehler lengthError zurück und breche diesen Algorithmus ab.~~
~~b. Schritt 2: Teile P auf in Blöcke mit jeweils 64 Bit $P == P_1 \| P_2 \| \ldots \| P_n$.~~
~~c. Schritt 3: Setze C_0 = I2OS(T_1, 8)~~
~~d. Schritt 4: Berechne C_i = 3TDES_ENC(K, C_{i-1} XOR P_i) für i = 1,…, n.~~
~~e. Schritt 5: Berechne C = $C_1 \| C_2 \| \ldots \| C_{n-1} \| C_n$.~~ **[<=]**

6.7.1.2 Verschlüsselung AES

A_13626 - (N004.000) K_COS

Tabelle 50: CosT_b34: Definition der Funktion AES_CBC_ENC(...)

Input:	P	Oktettstring, Klartext (*plaintext*), beliebiger Oktettstring beliebiger Länge, der verschlüsselt wird
	K	Beliebiger Oktettstring der Länge 16, 24 oder 32 Oktett, der als Schlüssel verwendet wird
	T_1	Beliebige nicht-negative Zahl, die als Startwert IV verwendet wird
Output:	C	Oktettstring, Chiffrat (*ciphertext*), das dieselbe Länge wie P besitzt
Errors:	*lengthError*	Die Länge von P ist kein ganzzahliges Vielfaches der Blocklänge
Notation:		C = AES_CBC_ENC(K, T_1, P)

Das COS MUSS C mittels K und T_1 gemäß [NIST sp800-38a#6.2] aus P berechnen.
Dabei sind folgende Aktionen durchzuführen:
a. Schritt 1: Wenn OctetLength(P) mod 16 ungleich 0 ist,
dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
b. Schritt 2: Teile P auf in Blöcke mit jeweils 128 Bit: $P == P_1 \| P_2 \| \ldots \| P_n$.
c. Schritt 3: Setze C_0 = I2OS(T_1, 16)
d. Schritt 4: Berechne C_i = AES_ENC(K, C_{i-1} XOR P_i) für i = 1,…, n.
e. Schritt 5: Berechne C = $C_1 \| C_2 \| \ldots \| C_{n-1} \| C_n$. **[<=]**

6.7.2 Symmetrische Entschlüsselung

Die symmetrische Entschlüsselung überführt ein Chiffrat *ciphertext* (Oktettstring beliebigen Inhalts und Länge) in einen Klartext *plaintext*. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine symmetrische Entschlüsselung als Funktion wie folgt verwendet:

6.7.2.1 Entschlüsselung 3TDES, Option_DES

(N004.100) ist absichtlich leer.

~~**Tabelle 51, CosT_ef0: Definition der Funktion 3TDES_CBX_DEC(...)**~~

~~Input:~~	~~C~~	~~Beliebiger Oktettstring, Chiffrat (*ciphertext*), der entschlüsselt wird, die Länge ist ein ganzzahliges Vielfaches der Blocklänge~~
	~~K~~	~~Beliebiger Oktettstring der Länge 24 Oktett, der als Schlüssel verwendet wird~~
	~~T_1~~	~~Beliebige nicht-negative Zahl, die als Startwert verwendet wird~~
~~Output:~~	~~P~~	~~Oktettstring, Klartext (plaintext)~~
~~Errors:~~	~~lengthError~~	~~Die Länge von C ist kein ganzzahliges Vielfaches der Blocklänge~~
~~Notation:~~		~~P = 3TDES_CBC_DEC(K, T_1, C)~~

A_13627 - (N004.100) K_COS_G1, Option_DES

~~Das COS MUSS P mittels K und T_1 wie folgt aus C berechnen. Dabei sind folgende Aktionen durchzuführen:~~
~~a. Schritt 0: Wenn OctetLength(P) mod 8 ungleich 0 ist,~~
~~dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.~~
~~b. Schritt 1: Teile C auf in Blöcke mit jeweils 64 Bit $C == C_1 \| C_2 \| \ldots \| C_n$.~~
~~c. Schritt 2: Setze C_0 = I2OS(T_1, 8)~~
~~d. Schritt 3: Berechne P_i = 3TDES_DEC(K, C_i) XOR C_{i-1} für i = 1,…, n.~~
~~e. Schritt 4: Berechne $P = P_1 \| P_2 \| \ldots \| P_{n-1} \| P_n$.~~**[<=]**

6.7.2.2 Entschlüsselung AES

A_13628 - (N004.200) K_COS

Tabelle 52: CosT_b49: Definition der Funktion AES_CBC_DEC(...)

Input:	C	Beliebiger Oktettstring, Chiffrat (*ciphertext*) beliebiger Länge, der entschlüsselt wird
	K	Beliebiger Oktettstring der Länge 16, 24 oder 32 Oktett, der als Schlüssel verwendet wird
	T_1	Beliebige nicht-negative Zahl, die als Startwert IV verwendet wird
Output:	P	Oktettstring, Klartext (*plaintext*), der dieselbe Länge wie C besitzt
Errors:	*lengthError*	Die Länge von C ist kein ganzzahliges Vielfaches der Blocklänge
Notation:		P = AES_CBC_DEC(K, T_1, C)

Das COS MUSS P mittels K und T_1 gemäß [NIST sp800-38a#6.2] aus C berechnen. Dabei sind folgende Aktionen durchzuführen:
a. Schritt 0: Wenn OctetLength(P) mod 16 ungleich 0 ist,
dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.

b. Schritt 1: Teile C auf in Blöcke mit jeweils 128 Bit: $C == C_1 \;||\; C_2 \;||\; \ldots \;||\; C_n$.
c. Schritt 2: Setze C_0 = I2OS(T_1, 16)
d. Schritt 3: Berechne P_i = AES_DEC(K, C_i) XOR C_{i-1} für i = 1,…, n.
e. Schritt 4: Berechne $P = P_1 \;||\; P_2 \;||\; \ldots \;||\; P_{n-1} \;||\; P_n$. **[<=]**

6.8 Vertraulichkeit von Daten, asymmetrischer Fall

6.8.1 Asymmetrische Verschlüsselung

Die asymmetrische Verschlüsselung überführt eine Nachricht M (Oktettstring beliebigen Inhalts und Länge) in ein Chiffrat C. Diese Funktionalität wird an der physikalischen Schnittstelle nicht unmittelbar sichtbar. Im Rahmen diverser interner Operationen im Betriebssystem wird eine asymmetrische Verschlüsselung als Funktion wie folgt verwendet:

6.8.1.1 RSA, ES, PKCS1 V1.5

Dieses Kapitel ist absichtlich leer.

(N004.300) Diese Anforderung ist absichtlich leer., weil die Verschlüsselung gemäß RSAES_PKCS1_V1_5_ENCRYPT ersatzlos entfallen ist.

6.8.1.2 RSA, OAEP, Verschlüsselung

G2_N004.400 - (N004.400) K_COS

Tabelle 53: CosT_089: Definition der Funktion RSAES_OAEP_ENCRYPT(...)

Input:	PuK	Öffentlicher RSA-Schlüssel gemäß (N009.100), (N00.9110) und (N009.200)
	M	Beliebiger Oktettstring, zu verschlüsselnde Nachricht
Output:	C	Oktettstring, Chiffrat zur Nachricht M
Errors:	ERROR	"message too long", falls M ist zu lang
Notation:		C = RSAES_OAEP_ENCRYPT(PuK, M)

Das COS MUSS C mittels PuK und M gemäß [PKCS#1] Kapitel 7.1.1 berechnen. Es gelten folgende Definitionen: n = $PuK.n$, e = $PuK.e$.
a. Schritt 1: Wenn OctetLength(M) größer als OctetLength(n) – 66 ist, dann breche diesen Algorithmus mit dem Fehler "ERROR" ab.
b. Schritt 2: Setze L = '', (Anmerkung: leerer Oktettstring).
c. Schritt 3: Setze $lHash$ = SHA_256(L).
d. Schritt 4: Setze $Plen$ = OctetLength(n) – OctetLength(M) – 66.
e. Schritt 5: Setze PS = I2OS(0, $Plen$).
f. Schritt 6: Setze DB = $lHash \;||\; PS \;||$ '01' $||\; M$.
g. Schritt 7: Setze $seed$ = RAND(32).
h. Schritt 8: Setze $dbMask$ = BS2OS(MGF(OS2BS($seed$), 8 (OctetLength(n) – 33), 0)).
i. Schritt 9: Setze $maskedDB$ = DB XOR $dbMask$.

j. Schritt 10: Setze $seedMask$ = BS2O2(MGF(OS2BS($maskedDB$), 8 OctetLength($lHash$), 0)).
k. Schritt 11: Setze $maskedSeed$ = $seed$ XOR $seedMask$.
l. Schritt 12: Setze EM = '00' || $maskedSeed$ || $maskedDB$.
m. Schritt 13: Setze m = OS2I(EM).
n. Schritt 14: Setze c = m^e mod n.
o. Schritt 15: Setze C = I2OS(c, OctetLength(n)).**[<=]**

6.8.1.3 Elliptic Curve Key Agreement

Die hier beschriebene Funktionalität ist an der physikalischen Schnittstelle nicht direkt sichtbar, wird aber im Rahmen verschiedener Funktionen mit elliptischen Kurven verwendet.

Hinweis CosH_c0c: In einer früheren Dokumentenversion wurde in diesem Kapitel die Funktion ECKA(...) spezifiziert. In dieser Dokumentenversion wurde die Funktion ECKA(...) aufgespalten in ECKApoint(...) und ECKAvalue(...).

6.8.1.3.1 Elliptic Curve Key Agreement Point Sab

Die hier beschriebene Funktionalität berechnet aus einem privaten und einem öffentlichen Schlüssel auf einer elliptischen Kurve einen Punkt, der als gemeinsames Geheimnis weiterverwendet wird.

Hinweis CosH_261: Dies entspricht dem ersten Teil der Funktion ECKA(...) aus früheren Dokumentenversionen. Der übrige Teil der Funktion ECKA(...) ist in (N004.492) enthalten.

G2_N004.490 - (N004.490) K_COS

Tabelle 54: CosT_eb0: Definition der Funktion ECKApoint(...)

Input:	d	natürliche Zahl, die dem privaten Schlüssel PrK entspricht
	P	Punkt auf derselben elliptischen Kurve, wie der private Schlüssel PrK
	dP	Domainparameter gemäß (N008.600)
Output:	S_{AB}	"gemeinsamer geheimer Punkt"
Errors:	ERROR	Wenn das Ergebnis in Schritt 3 der unendlich ferne Punkt ist
Notation:		S_{AB} = ECKApoint(d, P, dP)

Das COS MUSS S_{AB} mittels d, P und dP berechnen, wobei die Schritte 1 bis 3 [BSI-TR-03111#4.3.1] entsprechen. Es gelten folgende Definitionen:
h = $dP.h$, n = $dP.n$, L = $dP.L$
a. Schritt 1: l = h^{-1} mod n.
b. Schritt 2: Q = [h] P.
c. Schritt 3: S_{AB} = [$d\ l$ mod n] Q.
Wenn S_{AB} gleich dem unendlich fernen Punkt O der Kurve ist,
dann gebe den Fehler "ERROR" zurück und beende diesen Algorithmus,
sonst gebe S_{AB} zurück.

d. Dieser Punkt ist absichtlich leer. Er wurde nach (N004.492) verschoben.~~Schritt 4: K_{AB} = I2OS(x_S, L).~~ **[<=]**

6.8.1.3.2 Elliptic Curve Key Agreement Value Zab

Die hier beschriebene Funktionalität wandelt einen Punkt auf einer elliptischen Kurve, der ein gemeinsames Geheimnis darstellt, in einen Oktettstring um.

Hinweis CosH_35e: Dies entspricht dem zweiten Teil der Funktion ECKA(...) aus früheren Dokumentenversionen. Der erste Teil der Funktion ECKA(...) ist in (N004.490) enthalten.

A_17556 - (N004.492) K_COS

Tabelle 55: CosT_e3a: Definition der Funktion ECKAvalue(...)

Input:	d	natürliche Zahl, die dem privaten Schlüssel PrK entspricht
	P	Punkt auf derselben elliptischen Kurve, wie der private Schlüssel PrK
	dP	Domainparameter gemäß (N008.600)
Output:	Z_{AB}	"gemeinsamer geheimer Oktettstring"
Errors:	$ERROR$	Wenn die Funktion in Schritt 1 diesen Fehler meldet
Notation:		Z_{AB} = ECKAvalue(d, P, dP)

Das COS MUSS Z_{AB} mittels d, P und dP berechnen, wobei dies Schritt 4 in [BSI-TR-03111#4.3.1] entspricht.
a. Schritt 4.a: S_{AB} = ECKApoint(d, P, dP).
Wenn der Aufruf der Funktion ECKApoint(...) mit der Fehlermeldung $ERROR$ abbricht,
dann gebe $ERROR$ zurück und beende diesen Algorithmus,
sonst fahre mit Schritt 4.b fort.
b. Schritt 4.b: Z_{AB} = I2OS(x_s, $dp.L$) mit x_s == x-Komponente des Punktes S_{AB}.**[<=]**

6.8.1.4 ELC Verschlüsselung

G2_N004.500 - (N004.500) K_COS

Tabelle 56: CosT_a09: Definition der Funktion ELC_ENC(...)

Input:	M	Oktettstring, zu verschickende Nachricht mit beliebigem Inhalt und beliebiger Länge, die verschlüsselt wird
	PO_B	Oktettstring, öffentlicher Punkt P_B des Empfängers
	dP	Domainparameter gemäß (N008.600)
Output:	PO_A	Oktettstring, ephemer Punkt PE_A des Senders
	C	Oktettstring, Chiffrat der Nachricht M
	T	Oktettstring, MAC über das Chiffrat C

Errors:	*ERROR*	1. Wenn in Schritt 0 ein Fehler auftritt 2. Wenn in ECKAvalue(...) ein Fehler auftritt
Notation:		(PO_A, C, T) = ELC_ENC(M, PO_B, dP)

Das COS MUSS PO_A, C und T mittels M und PO_B berechnen, wobei ~~dies Schritt 3 in [BSI-TR-03111#4.3.1] entspricht und~~ folgende Definitionen gelten:
n = $dP.n$, h = $dP.h$, G = $dP.G$, L = $dP.L$
a. Schritt 0: P_B = OS2P(PO_B, dP).
Wenn diese Funktion mit einem Fehler terminiert,
dann gebe "ERROR" zurück und beende diesen Algorithmus.
b. Schritt 1: d = RNG({1, 2, …, n – 1}), d.h. zufällig erzeugte ganze Zahl aus dem Intervall [1, n – 1].
c. Schritt 2: PE_A = [d] G.
d. Schritt 3: K_{AB} = ECKAvalue(d, P_B, dP).
Wenn die Funktion ECKAvalue(...) einen Fehler meldet,
dann gebe *ERROR* zurück und beende diesen Algorithmus.
e. Schritt 4: Berechne abgeleitete Schlüssel gemäß (N001.520)
1. (K_{enc}, K_{mac}, T_2) = KeyDerivation_AES256(K_{AB}).
2. Setze: T_1 = OS2I(AES_ENC(K_{enc}, I2OS(T_2, 16)))
f. Schritt 5: PO_A = P2OS(PE_A, L).
g. Schritt 6: C = AES_CBC_ENC(K_{enc}, T_1, PaddingIso(M, 16)).
h. Schritt 7: T = CalculateCMAC_IsoPadding(K_{mac}, C).**[<=]**

6.8.2 Asymmetrische Entschlüsselung

Die asymmetrische Entschlüsselung überführt ein Chiffrat C in eine Nachricht M.

6.8.2.1 RSA, ES, PKCS1 V1.5, Decrypt

Dieses Kapitel ist absichtlich leer.

(N004.600) Diese Anforderung ist absichtlich leer.

6.8.2.2 RSA, OAEP, Decrypt

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Decipher sichtbar (siehe (N090.300)b).

G2_N004.700 - (N004.700) K_COS

Tabelle 57: CosT_c8a: Definition der Funktion RSAES_OAEP_DECRYPT(...)

Input:	*PrK*	Privater RSA-Schlüssel gemäß (N008.800)
	C	Beliebiger Oktettstring, Chiffrat der Nachricht *M*
Output:	*M*	Oktettstring, Klartextnachricht zum Chiffrat *C*
Errors:	*decryptionError*	falls einer der folgenden Fälle eintritt: *C* ist numerisch größer gleich dem Modulus von *PrK*
Notation:		*M* = RSAES_OAEP_DECRYPT(*PrK*, *C*)

Das COS MUSS M mittels PrK und C gemäß [PKCS#1#7.1.2] berechnen. Es gelten folgende Definitionen: n = $PrK.n$, d = $PrK.d$.
a. Schritt 1: Setze L = '', (Anmerkung: leerer Oktettstring).
b. Schritt 2: Setze c = OS2I(C).
c. Schritt 3: Wenn c größer gleich n ist,
dann breche diesen Algorithmus mit *decryptionError* ab.
d. Schritt 4: Setze m = c^d mod n.
e. Schritt 5: Setze EM = I2OS(m, OctetLength(n)).
f. Schritt 6: Setze $lHash$ = SHA_256(L).
g. Schritt 7: Teile EM wie folgt auf:
1. Schritt 7.1: EM == Y || $maskedSeed$ || $maskedDB$.
2. Schritt 7.2: 1 == OctetLength(Y).
3. Schritt 7.3: 32 == OctetLength($maskedSeed$).
h. Schritt 8: Setze $seedMask$ = BS2OS(MGF(OS2BS($maskedDB$), 8 OctetLength($lHash$), 0)).
i. Schritt 9: Setze $seed$ = $maskedSeed$ XOR $seedMask$.
j. Schritt 10: Setze $dbMask$ = BS2OS(MGF(OS2BS($seed$), 8 OctetLength($maskedDB$), 0)).
k. Schritt 11: Setze DB = $maskedDB$ XOR $dbMask$.
l. Schritt 12: Teile DB wie folgt auf:
1. Schritt 12.1: DB == $lHash'$ || PS || '01' || M.
2. Schritt 12.2: 32 == OctetLength($lHash'$).
3. Schritt 12.3: Der möglicherweise leere Oktettstring PS enthält nur Oktette mit dem Wert '00'.
m. Schritt 13: Breche diesen Algorithmus mit *decryptionError* ab, wenn
1. $lHash'$ ungleich $lHash$ ist, oder
2. Y nicht den Wert '00' besitzt, oder
3. kein Oktett mit dem Wert '01' existiert, welches PS von M trennt.
n. Gebe M zurück.**[<=]**

6.8.2.3 ELC Entschlüsselung

Diese Funktionalität wird an der physikalischen Schnittstelle im Rahmen des Kommandos PSO Decipher sichtbar (siehe (N090.300)c).

G2_N004.800 - (N004.800) K_COS

Tabelle 58: CosT_bb4: Definition der Funktion ELC_DEC(...)

Input:	PrK	Privater ELC–Schlüssel gemäß (N008.900) und (N009.000)
	PO	Oktettstring, ephemerer Punkt PE_A des Senders
	C	Oktettstring, Chiffrat der Nachricht M
	T'	Oktettstring, MAC über das Chiffrat C

Output:	M	Oktettstring, Klartextnachricht zum Chiffrat C
Errors:	$ERROR$	1. Wenn PO nicht im Format "uncompressed encoding" vorliegt 2. Wenn PO einen Punkt bezeichnet, der nicht auf derselben Kurve liegt, wie PrK 3. Wenn die Funktion ECKAvalue(...) einen Fehler meldet 4. Wenn die MAC-Prüfung fehlschlägt
Notation:		M = ELC_DEC(PO, PrK, C, T')

Das COS MUSS M mittels PE_A, PrK, C und T' berechnen, wobei ~~Schritt 1 [BSI-TR-03111#4.3.1] entspricht. Es gelten~~ folgende Definitionen gelten:
d = $PrK.d$, h = $PrK.domainParameter.h$, n = $PrK.domainParameter.n$, L = $PrK.domainParameter.L$

a. Schritt 0: PE_A = OS2P(PO, $PrK.domainParameter$).
Wenn diese Funktion mit einem Fehler terminiert,
dann gebe $ERROR$ zurück und beende diesen Algorithmus.
b. Schritt 1: K_{AB} = ECKAvalue(d, PE_A, $PrK.domainParameter$).
c. Schritt 2: Berechne abgeleitete Schlüssel gemäß (N001.520)
1. (K_{enc}, K_{mac}, T_2) = KeyDerivation_AES256(K_{AB})
2. Setze: T_1 = OS2I(AES_ENC (K_{enc}, I2OS(T_2, 16)))
d. Schritt 3: out = VerifyCMAC_IsoPadding(K_{mac}, T', C).
Wenn out den Wert $INVALID$ besitzt,
dann gebe $ERROR$ zurück und beende diesen Algorithmus.
e. Schritt 4: M = TruncateIso(AES_CBC_DEC(K_{enc}, T_1, C), 16)**[<=]**

7 CV-Zertifikat (informativ)

Die folgenden Punkte sind absichtlich leer: (N004.900), (N005.000), (N005.100), (N005.200), (N005.300), (N005.400), (N005.500), (N005.600), (N005.700), (N005.800), (N005.900), (N006.000), (N006.100), (N006.200), N006.300), (N006.400), (N006.500), (N006.600).

Bis zur Version 3.12.0 vom 15.05.2019 enthielten sie Anforderungen an CV-Zertifikate für RSA-Schlüssel. Seit dem 1.1.2019 werden derartige CV-Zertifikate in der Telematikinfrastruktur nicht mehr verwendet.

In [ISO/IEC 7816-8] werden zwei kompakte Zertifikatsformate definiert, sogenannte Card Verifiable Certificate (CVC), die eine Prüfung und Analyse durch eine Smartcard erleichtern. Typischerweise enthalten CV-Zertifikate nur die Informationen, welche eine Smartcard notwendigerweise für einen bestimmten Use Case benötigt. Sie sind deshalb kompakter als andere Zertifikatsformate, was sich positiv auf die Performanz auswirkt.

7.1 CV-Zertifikat für RSA-Schlüssel

In diesem Kapitel werden nur so genannte "nicht selbstbeschreibende" CV-Zertifikate betrachtet, welche eine Signatur mit "Message Recovery" gemäß [ISO/IEC 9796-2] DS1 enthalten.

Technisch betrachtet ist ein derartiges Zertifikat eine mittels [ISO/IEC 9796-2] DS1 signierte Nachricht M. Die Nachricht M enthält unterschiedliche Daten, die ohne Strukturinformationen oder Trennzeichen konkateniert sind, daher die Bezeichnung "nicht selbstbeschreibend". Damit die enthaltenen Informationen eindeutig extrahierbar sind, wird das höchstwertige Oktett von$M$ ausgewertet.

7.1.1 Bestandteile eines CV-Zertifikats für RSA-Schlüssel

Alle in 7.1 betrachteten signierten Nachrichten M enthalten die in den folgenden Unterkapiteln beschriebenen Informationen.

7.1.1.1 Certificate Profile Identifier (CPI)

Der Certificate Profile Identifier (CPI) zeigt die im CV-Zertifikat verwendete Struktur an.

7.1.1.2 Certification Authority Reference (CAR)

Die Certification Authority Reference (CAR) referenziert den Schlüssel der CA, welche das Zertifikat ausstellte. Typischerweise besitzt die CAR eine innere Struktur, welche sicherstellt, dass die CAR weltweit eineindeutig ist.

Typischerweise besitzt die ausstellende CA wiederum ein CV-Zertifikat. Auf diese Weise wird eine baumartige PKI aufgespannt mit der Root-CA an der Wurzel.

Der öffentliche Schlüssel der Root-CA wird oft als Sicherheitsanker bezeichnet und typischerweise in der Vorbereitungsphase (siehe Kapitel 4) in einer Smartcard

gespeichert. Deshalb ist es nicht notwendig, den Sicherheitsanker per Zertifikat zu importieren (Abbruch der Rekursion zum Zertifikatsimport).

7.1.1.3 Certificate Holder Reference (CHR)

Die Certificate Holder Reference (CHR) wird dazu verwendet, dem im Zertifikat enthaltenen öffentlichen Schlüssel einen eindeutigen Identifier zuzuordnen. Typischerweise wird die ebenfalls eindeutige Seriennummer einer Smartcard (ICCSN) als Teil der CHR verwendet. Im Rahmen dieses Dokumentes ist es irrelevant, wie die CHR gebildet wird.

7.1.1.4 Certificate Holder Autorisation (CHA)

Die Certificate Holder Autorisation (CHA) zeigt eine Rolle des Zertifikatsinhabers an. Typischerweise fordern Sicherheitskonzepte, dass zwei verschiedene Rollen unterhalb einer Root-CA nicht dieselbe CHA haben dürfen. Deshalb wird häufiger auch die Forderung erhoben, dass die CHA weltweit eineindeutig ist. Dann wird typischerweise ein Teil des eindeutigen Application Identifiers als Teil der CHA verwendet. Im Rahmen dieses Dokumentes ist es irrelevant, wie konkrete CHA-Werte festgelegt werden.

7.1.1.5 Object Identifier (OID)

Der Object Identifier (OID) in einem CVC beschreibt den Algorithmus, welcher dem Schlüssel im Zertifikat zugeordnet ist. Im Rahmen dieser Spezifikation werden verschiedene Algorithmen für verschiedene Verwendungszwecke genutzt, so dass implizit durch den Algorithmus-OID auch der Verwendungszweck des im Zertifikat enthaltenen öffentlichen Schlüssels festgelegt wird. OIDs sind weltweit eineindeutig.

Im Rahmen von 7.1 werden nur folgende OID-Werte verwendet (siehe CosT_a91) : - authS_ISO9796-2Withrsa_sha256_mutual und sigS_ISO9796-2Withrsa_sha256.

7.1.1.6 Öffentlicher Schlüssel

Der öffentliche RSA-Schlüssel besteht aus den in den folgenden Unterkapiteln beschriebenen Teilen.

7.1.1.6.1 Modulus

Der Modulus wird hexadezimal, vorzeichenlos im Big-Endian-Format codiert.

7.1.1.6.2 Öffentlicher Exponent

Der öffentliche Exponent wird hexadezimal, vorzeichenlos im Big-Endian-Format codiert.

7.1.2 Zertifikatsprofile für RSA-Schlüssel

Dieses Kapitel listet die zu unterstützenden Zertifikatsprofile auf, die anhand des CPI identifiziert werden.

7.1.2.1 CV-Zertifikat für CA-Schlüssel

In diesem Unterkapitel wird der Aufbau eines CV-Zertifikates beschrieben, welches den öffentlichen Schlüssel einer CA enthält. Dieser öffentliche Schlüssel ist wiederum

geeignet, im Rahmen eines PSO Verify Certificate-Kommandos (siehe 14.8.7.1) weitere öffentliche Schlüssel per CV-Zertifikat zu importieren. Für derartige CV-Zertifikate gilt:

A_13697 - (N004.900) K_externeWelt {K_Karte}, Option_RSA_CVC
Der CPI MUSS den Wert '21' haben.**[<=]**

A_13698 - (N005.000) K_externeWelt {K_Karte}, Option_RSA_CVC
Die OID MUSS den Wert sigS_ISO9796-2Withrsa_sha256 = '2B24 0304 0202 04' haben (siehe CosT_a91).**[<=]**

A_13699 - (N005.100) K_externeWelt {K_Karte}, Option_RSA_CVC
Die Länge des Modulus MUSS 2048 bit betragen.**[<=]**

A_13700 - (N005.200) K_externeWelt {K_Karte}, Option_RSA_CVC
Der öffentliche Exponent MUSS vier Oktett lang sein.**[<=]**

A_13701 - (N005.300) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CHR MUSS acht Oktett lang sein.**[<=]**

A_13702 - (N005.400) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CAR MUSS acht Oktett lang sein.**[<=]**

A_13703 - (N005.500) K_externeWelt {K_Karte}, Option_RSA_CVC
Für die zu signierende Nachricht M gilt:
M = CPI || Modulus || öffentlicherExponent || OID || CHR || CAR.**[<=]**

Wenn (N002.100)a berücksichtigt wird und der Hash-Wert mit einer Funktion aus 6.1 gebildet wird, dann gehören CHR und CAR stets zum "non recoverable part" der signierten Nachricht und liegen somit stets im Klartext vor.

7.1.2.2 CV-Zertifikat für Authentisierungsschlüssel

In diesem Unterkapitel wird der Aufbau eines CV-Zertifikats beschrieben, welches den öffentlichen Schlüssel einer Instanz enthält, welche diesen Schlüssel zu Authentisierungszwecken verwendet. Für derartige CV-Zertifikate gilt:

A_13704 - (N005.600) K_externeWelt {K_Karte}, Option_RSA_CVC
Der CPI MUSS den Wert '22' haben.**[<=]**

A_13705 - (N005.700) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CHA MUSS sieben Oktett lang sein.**[<=]**

A_13706 - (N005.800) K_externeWelt {K_Karte}, Option_RSA_CVC
Die OID MUSS den Wert authS_ISO9796-2Withrsa_sha256_mutual = '2B24 0305 0204' haben (siehe CosT_a91).**[<=]**

A_13707 - (N005.900) K_externeWelt {K_Karte}, Option_RSA_CVC
Die Länge des Modulus MUSS 2048 bit betragen.**[<=]**

A_13708 - (N006.000) K_externeWelt {K_Karte}, Option_RSA_CVC
Der öffentliche Exponent MUSS vier Oktett lang sein.**[<=]**

A_13709 - (N006.100) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CHR MUSS zwölf Oktett lang sein.**[<=]**

A_13710 - (N006.200) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CAR MUSS acht Oktett lang sein.**[<=]**

A_13711 - (N006.300) K_externeWelt {K_Karte}, Option_RSA_CVC

Für die zu signierende Nachricht M gilt:
M = CPI || Modulus || öffentlicherExponent || OID || CHA || CHR || CAR.[<=]

Wenn (N002.100)a berücksichtigt wird und der Hash-Wert mit einer Funktion aus 6.1 gebildet wird, dann gehören CHA, CHR und CAR stets zum "non recoverable part" der signierten Nachricht und liegen damit stets im Klartext vor.

7.1.3 Struktur und Inhalt eines CV-Zertifikats für RSA-Schlüssel

Die Inhalte eines Zertifikat-Datenobjektes werden wie folgt berechnet:

A_13712 - (N006.400) K_externeWelt {K_Karte}, Option_RSA_CVC
Schritt 1: Die Nachricht M MUSS gemäß (N005.500) oder (N006.300) erzeugt werden.[<=]

A_13713 - (N006.500) K_externeWelt {K_Karte}, Option_RSA_CVC
Schritt 2: Die Nachricht M MUSS mit einem privaten RSA-Schlüssel PrK signiert werden, dessen Moduluslänge 2048 bit beträgt. Als Signaturverfahren MUSS (N003.100) verwendet werden, so dass gilt:
($SIG.CA$, M_2) = RSA_ISO9796_2_DS1_SIGN(PrK, M).[<=]

A_13714 - (N006.600) K_Anwendungsspezifikation {K_Karte}, Option_RSA_CVC
Ein CV-Zertifikats-EF MUSS ein zusammengesetztes Zertifikat-Datenobjekt mit Tag = '7F21' (CV-Zertifikat) enthalten. Das Zertifikat-Datenobjekt MUSS genau zwei primitive Datenobjekte in der angegebenen Reihenfolge enthalten:
a. Datenelement $SIG.CA$ als Wertfeld in einem Datenobjekt mit Tag = '5F37'.
b. Non-recoverable part M_2 als Wertfeld in einem Datenobjekt mit Tag = '5F38'.[<=]

CV-Zertifikate für RSA-Schlüssel mit einer Moduluslänge von 2048 bit = 256 Oktett:

Tabelle 59, CosT_ad5: CV-Zertifikat einer CA mit CPI = '21', SHA-256

Tag	**L**	**Wert**		
'7F21'	'820146'	CV-Zertifikat ('0146' = 326 Oktett)		
		Tag	**L**	**Wert**
		'5F37'	'820100'	Signatur $SIG.CA$ ('0100' = 256 Oktett)
		'5F38'	'3E'	non recoverable part M_2 ('003E' = 62 Oktett)

~~**Tabelle 60, CosT_6ea: CV-Zertifikat zur Authentisierung mit CPI = '22', SHA-256**~~

~~**Tag**~~	~~**L**~~	~~**Wert**~~		
~~'7F21'~~	~~'820150'~~	~~CV-Zertifikat ('0150' = 336 Oktett)~~		
		~~**Tag**~~	~~**L**~~	~~**Wert**~~
		~~'5F37'~~	~~'820100'~~	~~Signatur *SIG.CA* ('0100' = 256 Oktett)~~
		~~'5F38'~~	~~'48'~~	~~non recoverable part M_2 ('0048' = 72 Oktett)~~

~~*Hinweis CosH_a67: Vorgaben für CV-Zertifikate für ELC-Schüssel finden sich in [gemSpec_PKI].*~~

7.2 CV-Zertifikate für ELC-Schlüssel

Normative Festlegungen zu CV-Zertifikaten für ELC-Schlüssel finden sich in [gemSpec_PKI].

8 Objekte

8.1 Diverse Attribute (normativ)

Dieses Unterkapitel beschreibt einige Attribute, die für mehrere Objekttypen gleichermaßen relevant sind.

8.1.1 File Identifier

Der Attributstyp *fileIdentifier* wird von den Objekttypen DF und Datei verwendet.

Aus der Norm [ISO/IEC 7816-4] leiten sich folgende Regeln ab, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N006.700 - (N006.700) K_Anwendungsspezifikation {K_Karte}
Der Wert von *fileIdentifier* MUSS eine ganze Zahl im Intervall ['1000', 'FEFF'] oder Element der Menge {'011C'} sein.**[<=]**

A_13717 - (N006.800) K_Anwendungsspezifikation {K_Karte}
Ein Objekt, das nicht *root* (siehe (N019.900)a) ist, DARF KEINEN *fileIdentifier* mit dem Wert '3F00' besitzen.**[<=]**

A_13718 - (N006.900) K_Anwendungsspezifikation {K_Karte}
KEIN Objekt DARF einen *fileIdentifier* mit dem Wert '3FFF' besitzen.**[<=]**

Hinweis CosH_596: Die Forderung (N006.700) ist strenger als [ISO/IEC 7816-4]. Dies lässt Raum für herstellerspezifische Ordner (DF) und Dateien (EF).

8.1.2 Short File Identifier

Es ist möglich, dass der Attributstyp *shortFileIdentifier* von den Objekttypen Datei verwendet wird (siehe 8.3.2).

Aus der Norm [ISO/IEC 7816-4] leiten sich folgende Regeln ab, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N007.000 - (N007.000) K_Anwendungsspezifikation {K_Karte}
Der Wert von *shortFileIdentifier* MUSS eine ganze Zahl im Intervall [1, 30] sein.**[<=]**

8.1.3 Life Cycle Status

Der Attributstyp *lifeCycleStatus* wird von den Objekttypen Objektsystem, Ordner, Datei, Rekord, Passwort und Schlüssel zur Speicherung eines "physikalischen" Life Cycle Status (siehe (N020.500)) verwendet. Zusätzlich ist im Rahmen gewisser Operationen der "logische" Life Cycle Status (siehe (N020.600)) wichtig. Bei der Kommandobeschreibung wird darauf verwiesen, ob der "physikalische" oder der "logische" Life Cycle Status zu verwenden ist.

Aus den Normen [ISO/IEC 7816-4#7.4.10] und [ISO/IEC 7816-9#Fig.1] leiten sich folgende Regeln ab, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13719 - (N007.100)a K_Anwendungsspezifikation {K_Karte}
Der Wert von *lifeCycleStatus* MUSS ein Element der Menge {
"Operational state (active)",
"Operational state (deactivated)",
"Termination state"
} sein.**[<=]**

A_13720 - (N007.100)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *lifeCycleStatus*
1. unterstützt oder
2. ablehnt.**[<=]**

8.1.4 Zugriffsregelliste

Der Attributstyp *interfaceDependentAccessRules* wird von den Objekttypen verwendet, welche die Ausführung von Kommandos von der Erfüllung von Zugriffsregeln abhängig machen. Dabei handelt es sich um eine Ansammlung von Zugriffsregeln. Typischerweise werden einem Objekt mehr als eine Zugriffsregel zugeordnet, weil es dem COS dadurch möglich ist, situationsbezogen zu reagieren. So sind gewisse Leseoperationen typischerweise nur über eine kontaktbehaftete Schnittstelle zulässig, während sie für eine kontaktlose Schnittstelle verboten sind. Andererseits ist der Zugriff auf deaktivierte Objekte typischerweise stark eingeschränkt. Darüber hinaus werden Security Environments genutzt, um Abhängigkeiten der Zugriffsregel von der Einsatzumgebung auszudrücken.

Akademisch betrachtet wäre eine dreidimensionale Matrix für *interfaceDependentAccessRules* angebracht, wo je nach Interface (kontaktbehaftet/kontaktlos = erste Dimension), Lebenszyklus (aktiviert/deaktiviert = zweite Dimension) und Einsatzumgebung (SE-Identifier = dritte Dimension) eine Zugriffsregel ausgewählt wird.

In dieser Dokumentenversion beschränkt sich die erste Dimension auf die beiden Fälle kontaktlose Schnittstelle gemäß 11.2.3 einerseits und kontaktbehaftete Schnittstelle gemäß 11.2.1 oder 11.2.2 andererseits und für die zweite Dimension ist es im deaktivierten oder terminierten Fall hinreichend, nur eine Einsatzumgebung vorzusehen. Deshalb werden in dieser Version des Dokumentes statt einer (mehrdimensionalen) Matrix zwei Listen verwendet.

A_13721 - (N007.170)a K_Anwendungsspezifikation {K_Karte}
Die Liste *interfaceDependentAccessRules* MUSS zwei Elemente umfassen.**[<=]**

G2_N007.170.b - (N007.170)b K_Anwendungsspezifikation {K_Karte}
Das erste Element der Liste MUSS eine *accessRuleList* gemäß (N007.200) bis (N007.500) sein und die Zugriffsregeln für eine kontaktbehaftete Kommunikation gemäß 11.2.1 und 11.2.2 beinhalten.**[<=]**

G2_N007.170.c - (N007.170)c K_Anwendungsspezifikation {K_Karte}
Das zweite Element der Liste MUSS eine *accessRuleList* gemäß (N007.200) bis (N007.500) sein und die Zugriffsregeln für eine kontaktlose Kommunikation gemäß 11.2.3 beinhalten.**[<=]**

A_13722 - (N007.170)d K_TST

~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS *interfaceDependentAccessRules* mit mehr Elementen
1. unterstützt oder
2. ablehnt.**[<=]**

A_13723 - (N007.170)e K_TST
~~Ein COS,~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS, welches die Option_kontaktlose_Schnittstelle nicht anbietet, ~~KANN~~ das zweite Listenelement in *interfaceDependentAccessRules*
1. beachtet oder
2. ignoriert.**[<=]**

G2_N007.200.a - (N007.200)a K_Anwendungsspezifikation {K_Karte}
Der Wert von *accessRuleList* MUSS eine Liste mit mindestens drei und maximal sechs Elementen sein.**[<=]**

A_13724 - (N007.200)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS *accessRuleList* mit mehr als sechs Elementen
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N007.300.a - (N007.300)a K_Anwendungsspezifikation {K_Karte}
Ein Listenelement in *accessRuleList* MUSS für den Wert der Variable *lifeCycleStatus* == "Operational state (deactivated)" (siehe (N007.100)a) verwendet werden.**[<=]**

G2_N007.300.b - (N007.300)b K_Anwendungsspezifikation {K_Karte}
Ein weiteres Listenelement in *accessRuleList* MUSS für den Wert der Variable *lifeCycleStatus* == "Termination state" (siehe (N007.100)a) verwendet werden.**[<=]**

G2_N007.400 - (N007.400) K_Anwendungsspezifikation {K_Karte}
Die übrigen Listenelemente in *accessRuleList* MÜSSEN genau einem *seIdentifier* Wert gemäß (N007.900) zugeordnet werden.**[<=]**

G2_N007.500 - (N007.500) K_Anwendungsspezifikation {K_Karte}
Jedes Listenelement in *accessRuleList* MUSS genau eine Zugriffsregel gemäß 10.3 enthalten. **[<=]**

8.1.5 Rekord

Der Objekttyp *record* wird von den Objekttypen linear variables, linear fixes und zyklisches Elementary File (EF) verwendet, siehe 8.3.2.2.

G2_N007.600.a - (N007.600)a K_Anwendungsspezifikation {K_Karte}
Ein *record* MUSS eine Rekordnummer *number* mit einem ganzzahligen Wert aus dem Intervall [1, 254] besitzen.**[<=]**

A_13725 - (N007.600)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS Rekordnummern mit weiteren Werten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N007.700.a - (N007.700)a K_Anwendungsspezifikation {K_Karte}
Ein *record* MUSS einen Oktettstring *data* mit einer Länge aus dem Intervall ['01', 'FF'] = [1, 255] besitzen.**[<=]**

A_13726 - (N007.700)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Längen für *record*
1. unterstützt oder
2. ablehnt.**[<=]**

A_13727 - (N007.800) K_Anwendungsspezifikation {K_Karte}
Ein *record* MUSS eine Liste mit Attributen vom Typ *lifeCycleStatus* (siehe (N007.100)a) unterstützen.**[<=]**

G2_N007.800.a - (N007.800)a K_Anwendungsspezifikation {K_Karte}
Die Liste *lifeCycleStatus* MUSS entweder leer sein (der *record* hat keinen *lifeCycleStatus* und befindet sich implizit im Zustand "Operational state (active)") oder**[<=]**

G2_N007.800.b - (N007.800)b K_Anwendungsspezifikation {K_Karte}
die Liste *lifeCycleStatus* MUSS genau ein Element mit einem Wert aus der Menge {"Operational state (active)", "Operational state (deactivated)"} enthalten (der *record* hat genau einen *lifeCycleStatus*).**[<=]**

Hinweis CosH_d67: Da es aus funktionaler Sicht keine Möglichkeit gibt den lifeCycleStatus *eines* record *in den Zustand "Termination state" zu überführen, ist dieser Zustand für einen* record *irrelevant.*

8.1.6 SE-Identifier

Der Attributstyp *seIdentifier* ist eng mit dem Begriff Security Environment verknüpft (siehe 8.8). Aus dem gemäß [ISO/IEC 7816-4#10.3.3] erlaubten Wertebereich wird hier folgende Untermenge ausgewählt:

G2_N007.900.a - (N007.900)a K_Anwendungsspezifikation {K_Karte}
Der Wert von *seIdentifier* MUSS eine ganze Zahl im Intervall [1, 4] sein.**[<=]**

A_13728 - (N007.900)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *seIdentifier*
1. akzeptiert oder
2. ablehnt.**[<=]**

8.1.7 PIN

Der Attributstyp *pin* wird im Zusammenhang mit der Benutzerverifikation verwendet. Es gilt:

A_13729 - (N008.000)a K_Anwendungsspezifikation {K_Karte}
Ein *pin* MUSS eine Folge von Ziffern aus dem Wertebereich {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} mit einer Länge aus dem Intervall [4, 12] sein.**[<=]**

A_13732 - (N008.000)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Zeichen oder weitere Längen für *pin*
1. unterstützt oder
2. ablehnt.**[<=]**

A_13733 - (N008.100) K_Anwendungsspezifikation {K_Karte}

Für die Umwandlung des Attributtyps *pin* nach Format-2-PIN-Block MUSS gelten:
a. Der Format-2-PIN-Block ist ein Oktettstring mit acht Oktett = 16 Nibble.
b. Das erste Nibble hat den Wert '2'.
c. Das zweite Nibble codiert hexadezimal die Anzahl Ziffern in *pin*.
d. Das *i*+2–te Nibble codiert hexadezimal die *i*–te Ziffer von *pin*.
e. Alle anderen Nibble besitzen den Wert 'F'.**[<=]**

8.1.8 Datum

Der Attributstyp *date* wird im Zusammenhang mit dem Gültigkeitszeitraum von Schlüsseln verwendet.

A_13734 - (N008.120)a K_Anwendungsspezifikation {K_Karte}
Ein *date* MUSS ein Datum in der Form YYMMDD in unkomprimierter BCD-Form enthalten. Beispiel: '010400050203' = 140523 = 23. Mai 2014. COS intern wird *date* als natürliche Zahl interpretiert.**[<=]**

A_13735 - (N008.120)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Codierungen für *date*
1. unterstützt oder
2. ablehnt.**[<=]**

8.2 Schlüsselmaterial (normativ)

8.2.1 Symmetrische Schlüssel

8.2.1.1 3TDES-Schlüssel, Option_DES

(N008.200) ist absichtlich leer.
(N008.300) ist absichtlich leer.

~~Der Attributstyp *3TDES_Key* dient der Speicherung von Schlüsselmaterial für einen 3TDES-Schlüssel, welcher aus drei Teilschlüsseln *Ka*, *Kb* und *Kc* besteht.~~

G2_N008.200 - (N008.200) K_Anwendungsspezifikation {K_Karte}, Option_DES
~~Der Wert von *3TDES_Key* MUSS ein Oktettstring mit 24 Oktetten sein.~~**[<=]**

A_13736 - (N008.300)a K_Anwendungsspezifikation {K_Karte}, Option_DES
~~Der Wert von *3TDES_Key* MUSS beliebig wählbar sein~~**[<=]**

A_13737 - (N008.300)b K_TST, Option_DES
~~Die funktionale Eignung DARF einem Prüfling NICHT mit der Begründung verwehrt werden eine inkorrekte Parität eines Oktettes in *3TDES_Key* würde~~
~~1. akzeptiert oder~~
~~2. abgelehnt.~~**[<=]**

~~*Hinweis CosH_654: Aus Sicherheitsgründen sind die Teilschlüssel Ka, Kb und Kc eines 3TDES-Schlüssels paarweise verschieden zu wählen. Diese Anforderung richtet sich nicht an das COS, sondern an die externe Welt, welche derartige Schlüssel etwa im Rahmen der Personalisierung in die Smartcard einbringt.*~~

~~*Hinweis CosH_afc: Aus Sicherheitsgründen sind Werte für die Teilschlüssel Ka, Kb oder Kc eines 3TDES-Schlüssels zu vermeiden, die einem schwachen oder halbschwachen DES-Schlüssel*~~

~~*entsprechen (siehe Glossar). Diese Anforderung richtet sich nicht an das COS, sondern an die externe Welt, welche derartige Schlüssel etwa im Rahmen der Personalisierung in die Smartcard einbringt.*~~

8.2.1.2 AES-128-Schlüssel

Der Attributstyp *aes128Key* dient der Speicherung von Schlüsselmaterial für einen AES-128-Schlüssel.

G2_N008.400 - (N008.400) K_Anwendungsspezifikation {K_Karte}
Der Wert von *aes128Key* MUSS ein Oktettstring mit sechzehn Oktette sein.**[<=]**

G2_N008.500 - (N008.500) K_COS
Das COS MUSS beliebige Werte für *aes128Key* unterstützen.**[<=]**

8.2.1.3 AES-192-Schlüssel

Der Attributstyp *aes192Key* dient der Speicherung von Schlüsselmaterial für einen AES-192-Schlüssel.

G2_N008.520 - (N008.520) K_Anwendungsspezifikation {K_Karte}
Der Wert von *aes192Key* MUSS ein Oktettstring mit 24 Oktette sein.**[<=]**

G2_N008.525 - (N008.525) K_COS
Das COS MUSS beliebige Werte für *aes192Key* unterstützen.**[<=]**

8.2.1.4 AES-256-Schlüssel

Der Attributstyp *aes256Key* dient der Speicherung von Schlüsselmaterial für einen AES-256-Schlüssel.

G2_N008.540 - (N008.540) K_Anwendungsspezifikation {K_Karte}
Der Wert von *aes256Key* MUSS ein Oktettstring mit 32 Oktette sein.**[<=]**

G2_N008.545 - (N008.545) K_COS
Das COS MUSS beliebige Werte für *aes256Key* unterstützen.**[<=]**

8.2.2 Domainparameter für elliptische Kurven

Der Attributstyp *domainParameter* dient der Speicherung von Parametern, welche eine elliptische Kurve charakterisieren. Konform zu [BSI-TR-03111#Tabelle 2.1] werden die Domainparameter in diesem Dokument wie folgt dargestellt:

Tabelle 61: CosT_92d: Liste der Domainparameter einer elliptischen Kurve

Parameter	Bedeutung
p	Primzahl, welche die zugrunde liegende Gruppe $\mathrm{F_p}$ beschreibt
a	Erster Koeffizient der Weierstraßschen Gleichung
b	Zweiter Koeffizient der Weierstraßschen Gleichung
G	Ein Punkt auf der Kurve $\mathrm{E(F_p)}$, Basispunkt
n	Ordnung des Basispunktes G in $\mathrm{E(F_p)}$
h	Cofaktor von G in $\mathrm{E(F_p)}$, wegen (N002.500) gilt h = 1 für alle verpflichtend zu unterstützenden Kurven
L	siehe (N008.600)b
t	siehe (N008.600)c
OID	Object Identifier, der die elliptische Kurve referenziert, siehe (N008.600)d

Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten, vergleiche CosT_92d:

A_13747 - (N008.600) K_COS
Der Attributstyp *domainParameter* MUSS enthalten

1. alle Parameter aus ~~Tabelle 7~~ [BSI-TR-03111#Tabelle 2.1] und
2. eine Zahl L, welche die minimale Anzahl Oktette angibt, die nötig sind, um p als vorzeichenlose Zahl zu codieren. Dieser Parameter wurde ~~der "offiziellen" Liste der Domainparameter~~ [BSI-TR-03111#Tabelle 2.1] hinzugefügt, da dieser Parameter in diesem Dokument vielfach verwendet wird. Allgemein gilt L=ceiling($\log_{256} p$). Wegen (N002.500) gilt:
   1. L = 32 für brainpoolP256r1 und ansix9p256r1.
   2. L = 48 für brainpoolP384r1 und ansix9p384r1.
   3. L = 64 für brainpoolP512r1
3. eine Zahl t, welche die Bitlänge von n angibt (siehe [BSI-TR-03111#Table 1.1], dort mit dem griechischen Buchstaben tau bezeichnet). Dieser Parameter wurde ~~der "offiziellen" Liste der Domainparameter~~ [BSI-TR-03111#Tabelle 2.1] hinzugefügt, da dieser Parameter in diesem Dokument vielfach verwendet wird. Allgemein gilt t=ceiling($\log_2 n$). Wegen (N002.500) gilt:
   1. t = 256 für brainpoolP256r1 und ansix9p256r1.
   2. t = 384 für brainpoolP384r1 und ansix9p384r1.
   3. t = 512 für brainpoolP512r1.
4. einen Oktettstring OID, durch den die Domainparameter (p, a, b, G, n, h) weltweit eindeutig bestimmt werden. Dieser Parameter wurde ~~der "offiziellen" Liste der Domainparameter~~ [BSI-TR-03111#Tabelle 2.1] hinzugefügt, da in diesem Dokument Domainparameter vielfach per OID referenziert werden. OID MUSS so aus CosT_a91 gewählt werden, dass damit eine elliptische Kurve gemäß (N002.500) referenziert wird.

[<=]

8.2.3 Privater Schlüssel

Der Attributstyp *privateKey* dient als Oberbegriff für *privateRsaKey* und *privateElcKey*.

8.2.3.1 Privater RSA-Schlüssel

Der Attributstyp *privateRsaKey* dient der Speicherung des privaten Teils eines asymmetrischen RSA-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

A_13748 - (N008.700) K_Anwendungsspezifikation {K_Karte}
Ein *privateRsaKey* MUSS ein Attribut Modulus n mit einer Länge gemäß (N002.100)a besitzen.**[<=]**

A_13750 - (N008.710) K_Anwendungsspezifikation {K_Karte}
Ein *privateRsaKey* MUSS ein Attribut *modulusLength* gemäß (N002.100)a besitzen.**[<=]**

A_13751 - (N008.800) K_Anwendungsspezifikation {K_Karte}
Ein *privateRsaKey* MUSS die im Folgenden genannten Attribute besitzen wobei gilt:
a. Wenn das Attribut *keyAvailable* (siehe (N018.200)) den Wert
   1. False besitzt, dann besitzen alle Attribute den Wert `AttributNotSet`.
   2. True besitzt, dann sind alle Attribute ganze Zahlen mit den unten genannten Eigenschaften.
b. Attribut Modulus n mit $n = p\,q$.
c. Attribut öffentlicher Exponent e mit $\gcd(e, (p-1)(q-1)) = 1$.
d. Attribut privater Exponent d mit $e\,d \equiv 1 \bmod \mathrm{lcm}(p-1, q-1)$.
e. Attribut Primzahl p mit p ist prim und $p < q$.
f. Attribut Primzahl q mit q ist prim.
g. Attrtibut CRT Exponent d_P mit $d_P = d \bmod (p-1)$.
h. Attrtibut CRT Exponent d_q mit $d_q = d \bmod (q-1)$.
i. Attribut c mit $c = q^{-1} \bmod p$. **[<=]**

8.2.3.2 Privater ELC-Schlüssel

Der Attributstyp *privateElcKey* dient der Speicherung des privaten Teils eines asymmetrischen ELC-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

A_13752 - (N008.900) K_Anwendungsspezifikation {K_Karte}
Ein *privateElcKey* MUSS ein Attribut *domainParameter* gemäß (N008.600) besitzen, das gemäß (N008.600)d per *OID* referenzierbar ist.**[<=]**

A_13753 - (N009.000) K_Anwendungsspezifikation {K_Karte}
Ein *privateElcKey* MUSS ein Attribut d mit einem ganzzahligen Wert aus dem Intervall [1, *domainParameter.n* – 1] besitzen.**[<=]**

8.2.4 Öffentlicher Schlüssel

Der Attributstyp *publicKey* dient als Oberbegriff für *publicRsaKey* und *publicElcKey*.

8.2.4.1 Öffentlicher RSA-Schlüssel

Der Attributstyp *publicRsaKey* dient der Speicherung des öffentlichen Teils eines asymmetrischen RSA-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

A_13755 - (N009.100) K_Anwendungsspezifikation {K_Karte}
Ein *publicRsaKey* MUSS ein Attribut Modulus n mit einer Länge gemäß (N002.100)a besitzen**[<=]**

A_13757 - (N009.110) K_Anwendungsspezifikation {K_Karte}
Ein *publicRsaKey* MUSS ein Attribut *modulusLength* gemäß (N002.100)a besitzen.**[<=]**

A_13758 - (N009.200) K_Anwendungsspezifikation {K_Karte}
Ein *publicRsaKey* MUSS ein Attribut öffentlicher Exponent e gemäß (N002.100)b besitzen.**[<=]**

8.2.4.2 Öffentlicher ELC-Schlüssel

Der Attributstyp *publicElcKey* dient der Speicherung des öffentlichen Teils eines asymmetrischen ELC-Schlüsselpaares. Bei der Spezifikation von Anwendungen sind folgende Regeln zu beachten:

A_13759 - (N009.300) K_Anwendungsspezifikation {K_Karte}
Ein *publicElcKey* MUSS ein Attribut *domainParameter* gemäß (N008.600) besitzen, das gemäß (N008.600)d per *OID* referenzierbar ist.**[<=]**

A_13760 - (N009.400) K_Anwendungsspezifikation {K_Karte}
Ein *publicElcKey* MUSS ein Attribut P besitzen, das einen vom unendliche fernen Punkt O verschiedenen Punkt auf der durch die Domainparameter vorgegebenen Kurve bezeichnet.**[<=]**

8.2.5 Transportschutz für ein Passwort

Der Attributstyp *transportStatus* gibt an, ob ein Passwort mit einem Transportschutz versehen ist. Der Wert dieses Attributes gibt also an, ob ein VERIFY Kommando möglich ist (siehe (N082.800)) und welche Variante von CHANGE REFERENCE DATA anzuwenden ist. An der externen Schnittstelle des COS wird dieser Attributstyp im Rahmen des Kommandos GET PIN STATUS verwendet (siehe CosT_a97).

G2_N009.500 - (N009.500) K_Anwendungsspezifikation {K_Karte}
Für *transportStatus* gilt folgender Wertebereich:
a. regularPassword
b. Leer-PIN
c. Transport–PIN**[<=]**

A_13762 - (N009.600)a K_COS
Das COS MUSS für *transportStatus* alle Werte aus (N009.500) unterstützen.**[<=]**

A_18243 - (N009.600)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für transportStatus
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N009.700 - (N009.700) K_externeWelt {K_Karte}

Wenn zur Aufhebung des Transportschutzes das Kommando CHANGE REFERENCE DATA verwendet wird (siehe 14.6.1), dann MUSS in Abhängigkeit des Wertes von *transportStatus* an der physikalischen Schnittstelle eine Variante gemäß CosT_16b verwendet werden.**[<=]**

A_13763 - (N009.710) K_COS
Wenn *oldSecret* oder *newSecret* in CosT_16b als "beliebig" gekennzeichnet ist, so MUSS deren Inhalt im Rahmen von (N008.000) und (N008.100) beliebig wählbar sein.

Tabelle 62: CosT_16b: Aufheben des Transportschutzes

Transportschutz	Parameter für CHANGE REFERENCE DATA
regularPassword	P1 = '00',*oldSecret* und *newSecret* beliebig
Leer-PIN	P1 = '01', *oldSecret* = '', das heißt leerer String,*newSecret* beliebig
Transport-PIN	P1 = '00',*oldSecret* und *newSecret* beliebig

[<=]

Hinweis CosH_ab5: Auch das Kommando RESET RETRY COUNTER ist in der Lage, den Transportschutz aufzuheben (siehe Use Cases in (N079.300) und (N080.100), sowie (N081.200)b).

8.3 File (normativ)

Dieses Unterkapitel beschreibt file-orientierte Objekttypen. File wird in diesem Zusammenhang als Oberbegriff für Ordner (siehe 8.3.1) und Datei (siehe 8.3.2) verwendet.

8.3.1 Ordner

Ordner dienen der hierarchischen Anordnung von Objekten in einem Objektsystem. In diesem Dokument wird Ordner als Oberbegriff für

- Applikationen (siehe 8.3.1.1),
- Dedicated Files (siehe 8.3.1.2) und
- Application Dedicated Files (siehe 8.3.1.3)

verwendet. Gemäß der Norm [ISO/IEC 7816-4] gelten für einen Ordner folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N009.800 - (N009.800) K_Anwendungsspezifikation {K_Karte}
Ein Ordner MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100)a) besitzen.**[<=]**

G2_N009.850 - (N009.850)a K_Anwendungsspezifikation {K_Karte}, Option_logische_Kanäle
Ein Ordner MUSS genau ein Attribut *shareable* vom Typ Boolean mit einem Wert True besitzen, wodurch angezeigt wird, dass dieser Ordner in mehr als einem logischen Kanal als *currentFolder* verwendbar ist (siehe (N029.900)a).**[<=]**

A_13765 - (N009.860) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *shareable*
a unterstützt oder
b. ablehnt.**[<=]**

G2_N009.900 - (N009.900) K_Anwendungsspezifikation {K_Karte}
Ein Ordner MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

A_13771 - (N010.000) K_Anwendungsspezifikation {K_Karte}
Ein Ordner MUSS eine (möglicherweise leere) Liste *children* mit Kindobjekten besitzen.**[<=]**

G2_N010.000.a.1 - (N010.000)a.1 K_Anwendungsspezifikation {K_Karte}
Unter Berücksichtigung der maximalen Schachtelungstiefe (siehe (N020.390)~~(N020.400)~~) MÜSSEN Listenelemente der folgenden Objekttypen unterstützt werden:
i. Applikation (siehe 8.3.1.1)
ii. Dedicated File (siehe 8.3.1.2)
iii. Application Dedicated File (siehe 8.3.1.3)
iv. Datei (siehe 8.3.2)
v. Reguläres Passwort (siehe 8.4)
vi. Multireferenz-Passwort (siehe 8.5)
vii. symmetrisches Authentisierungsobjekt (siehe 8.6.1)
viii. Privates Schlüsselobjekt (siehe 8.6.3) **[<=]**

G2_N010.000.a.2 - (N010.000)a.2 K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle
Das Listenelement Symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) MUSS unterstützt werden.**[<=]**

A_13772 - (N010.000)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Objekttypen als Listenelemente
1. unterstützt oder
2. ablehnt.**[<=]**

A_13773 - (N010.000)c K_Anwendungsspezifikation {K_Karte}
Für alle Elemente der Liste *children*, welche die Kindobjekte eines Ordners enthält, gilt: In der List *children* eines Ordners DARF es KEINE Kindobjekte geben,
1. deren Attribut *fileIdentifier* (sofern vorhanden, siehe (N010.600) und (N010.800)) denselben Wert besitzt.
2. deren Attribut *shortFileIdentifier* (sofern vorhanden, siehe (N010.900)a) denselben Wert besitzt.
3. deren Attribut *pwdIdentifier* (sofern vorhanden, siehe (N015.000)a) denselben Wert besitzt.
4. deren Attribut *keyIdentifier* (sofern vorhanden, siehe (N016.400)a, (N017.020)a und (N017.100)a) denselben Wert besitzt.**[<=]**

Bevor die folgenden Kommandos in der Lage sind mit dem Ordner zu arbeiten, ist er zu selektieren. Dies geschieht mittels SELECT-Kommando (siehe 14.2.6).

G2_N010.005.a - (N010.005)a K_COS
Das COS MUSS für Ordner das Kommandos ACTIVATE (siehe (N034.800)) unterstützen.**[<=]**

G2_N010.005.b - (N010.005)b K_COS
Das COS MUSS für Ordner das Kommando DEACTIVATE (siehe (N036.000)) unterstützen.**[<=]**

A_13774 - (N010.005)c K_COS
Das COS MUSS für Ordner das Kommando DELETE (siehe (N037.100)) unterstützen.**[<=]**

A_13775 - (N010.005)d K_COS
Das COS MUSS für Ordner das Kommando FINGERPRINT (siehe (N096.454)) unterstützen.**[<=]**

A_13776 - (N010.005)e K_COS
Das COS MUSS für Ordner das Kommando GET RANDOM (siehe (N099.322)) unterstützen.**[<=]**

A_13777 - (N010.005)f K_COS
Das COS MUSS für Ordner das Kommando LOAD APPLICATION (siehe (N038.600) und (N038.800)) unterstützen.**[<=]**

A_13778 - (N010.005)g K_COS
Das COS MUSS für Ordner das Kommando SELECT (siehe 14.2.6) unterstützen.**[<=]**

A_13779 - (N010.005)h K_COS
Das COS MUSS für Ordner das Kommando TERMINATE DF (siehe (N048.800)) unterstützen.**[<=]**

A_13921 - (N010.010) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für Ordner weitere Kommandos
a. unterstützt oder
b. ablehnt.**[<=]**

Einem Ordner sind weitere, kanalspezifische Attribute zugeordnet, die typischerweise einem flüchtig (etwa im RAM) gespeicherten Kanalkontext (siehe (N030.000)) zugerechnet werden. Sie werden typischerweise im Rahmen einer Selektion des Ordners verändert.

8.3.1.1 Applikation

Eine Applikation ist ein Ordner, der nur mittels AID selektierbar ist. Dies ist der Hauptunterschied zu einem DF (siehe 8.3.1.2).

Gemäß der Norm [ISO/IEC 7816-4] gelten für eine Applikation folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13844 - (N010.100) K_Anwendungsspezifikation {K_Karte}
Eine Applikation ist eine Erweiterung von Ordner und MUSS deshalb den Anforderungen aus 8.3.1 genügen.**[<=]**

A_13845 - (N010.200) K_Anwendungsspezifikation {K_Karte}
Das Attribut *applicationIdentifier* (AID) MUSS ein Oktettstring sein.**[<=]**

G2_N010.200.a - (N010.200)a K_Anwendungsspezifikation {K_Karte}
Die Länge von *applicationIdentifier* MUSS im Intervall [5, 16] liegen.**[<=]**

A_13846 - (N010.200)b K_COS
Die Oktette in *applicationIdentifier* MÜSSEN beliebig wählbar sein.**[<=]**

A_13847 - (N010.200)c K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS *applicationIdentifier*, welche gegen die in (N010.200){a, b} genannten Regeln verstoßen,
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N010.300 - (N010.300) K_Anwendungsspezifikation {K_Karte}
Eine Applikation MUSS mindestens einen *applicationIdentifier* besitzen.**[<=]**

G2_N010.400.a - (N010.400)a K_Anwendungsspezifikation {K_Karte}
Eine Applikation DARF NICHT mehr als zwei *applicationIdentifier* besitzen.**[<=]**

A_13850 - (N010.400)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS mehr als die in (N010.400)a genannte Anzahl an *applicationIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

8.3.1.2 Dedicated File

Ein Dedicated File (DF) ist ein Ordner, der nur mittels *fileIdentifier* selektierbar ist. Dies ist der Hauptunterschied zu einer Applikation (siehe 8.3.1.1).Gemäß der Norm [ISO/IEC 7816-4] gelten für ein DF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13851 - (N010.500) K_Anwendungsspezifikation {K_Karte}
Ein DF ist eine Erweiterung von Ordner und MUSS deshalb den Anforderungen aus 8.3.1 genügen.**[<=]**

G2_N010.600 - (N010.600) K_Anwendungsspezifikation {K_Karte}
Ein DF MUSS genau ein Attribut vom Typ *fileIdentifier* (siehe 8.1.1) besitzen.**[<=]**

8.3.1.3 Application Dedicated File

Ein Application Dedicated File (ADF) ist ein Ordner, der sowohl mittels *applicationIdentifier* als auch *fileIdentifier* selektierbar ist. ADF wird in diesem Dokument also als "Vereinigungsmenge" von Applikation (siehe8.3.1.1) und DF (siehe 8.3.1.2) gesehen. Dementsprechend gelten für ein ADF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_17454 - (N010.700) K_Anwendungsspezifikation {K_Karte}
Ein ADF MUSS die Eigenschaften von Applikation und DF in sich vereinen. Deshalb MUSS es sowohl die Anforderungen aus 8.3.1.1 als auch aus 8.3.1.2 erfüllen.**[<=]**

8.3.2 Datei

Eine Datei dient in diesem Dokument als Oberbegriff für transparente EF (siehe 8.3.2.1) und strukturierte EF (siehe 8.3.2.2).

Gemäß der Norm [ISO/IEC 7816-4] gelten für eine Datei folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N010.800 - (N010.800) K_Anwendungsspezifikation {K_Karte}
Eine Datei MUSS genau ein Attribut vom Typ *fileIdentifier* (siehe 8.1.1) besitzen.**[<=]**

A_13852 - (N010.900) K_COS

Eine Datei MUSS eine Liste mit Elementen vom Typ *shortFileIdentifier* (siehe (N007.000)) unterstützen.**[<=]**

G2_N010.900.a - (N010.900)a K_Anwendungsspezifikation {K_Karte}
Die Liste mit *shortFileIdentifier* MUSS entweder leer sein (die Datei hat keinen *shortFileIdentifier*), oder**[<=]**

G2_N010.900.b - (N010.900)b K_Anwendungsspezifikation {K_Karte}
die Liste mit *shortFileIdentifier* enthält genau ein Element (die Datei hat genau einen *shortFileIdentifier*).**[<=]**

G2_N011.000 - (N011.000) K_Anwendungsspezifikation {K_Karte}
Eine Datei MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100a)) besitzen.**[<=]**

G2_N011.050 - (N011.050) K_Anwendungsspezifikation {K_Karte}, Option_logische_Kanäle
Eine Datei MUSS genau ein Attribut *shareable* vom Typ Boolean mit dem Wert True besitzen, wodurch angezeigt wird, dass diese Datei in mehr als einem logischen Kanal als *currentEF* verwendbar ist (siehe (N029.900)m]).**[<=]**

A_17582 - (N011.060) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *shareable*
a unterstützt oder
b. ablehnt.**[<=]**

G2_N011.100 - (N011.100) K_Anwendungsspezifikation {K_Karte}
Eine Datei MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

A_13855 - (N011.200)a K_Anwendungsspezifikation {K_Karte}
Eine Datei MUSS ein Attribut *flagTransactionMode* vom Typ Boolean besitzen.**[<=]**

A_13856 - (N011.200)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS pro Datei ein solches Attribut individuell verwaltet.
[<=]

A_13857 - (N011.200)c K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS dieses Attribut für alle Dateien implizit auf True setzt.**[<=]**

A_13858 - (N011.300)a K_Anwendungsspezifikation {K_Karte}
Eine Datei MUSS ein Attribut *flagChecksum* vom Typ Boolean besitzen.**[<=]**

A_13859 - (N011.300)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS pro Datei ein solches Attribut individuell verwaltet.
[<=]

A_13860 - (N011.300)c K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS dieses Attribut für alle Dateien implizit auf True setzt.**[<=]**

Hinweis CosH_249: Das Setzen des Flags flagTransactionMode *verringert die Performanz bei Schreibzugriffen.*

Hinweis CosH_ad4: Das Setzen des Flags flagChecksum *verringert die Performanz bei Schreib- und Lesezugriffen.*

8.3.2.1 Transparentes Elementary File

Ein transparentes Elementary File (transparentes EF) dient der Speicherung eines Oktettstrings, wobei beliebige Teile des Oktettstrings zugreifbar sind. Der Inhalt von Oktettstrings in transparenten EF, welche im Rahmen einer Anwendungsspezifikation definiert werden, wird von einem COS lediglich gespeichert, niemals aber interpretiert oder für karteninterne Prozesse verwendet. Der Oktettstring wird in "data units" unterteilt (siehe [ISO/IEC 7816-4]). Die "data units" werden über einen Offset referenziert.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein transparentes EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13861 - (N011.400) K_Anwendungsspezifikation {K_Karte}
Ein transparentes EF ist eine Erweiterung von Datei und MUSS deshalb den Anforderungen aus 8.3.2 genügen.**[<=]**

G2_N011.500.a - (N011.500)a K_Anwendungsspezifikation {K_Karte}
Ein transparentes EF MUSS genau ein Attribut *numberOfOctet* mit einem ganzzahligen Wert aus dem Intervall [1, 32.768] besitzen.**[<=]**

A_13863 - (N011.500)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *numberOfOctet*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N011.510 - (N011.510) K_Anwendungsspezifikation {K_Karte}
Ein transparentes EF MUSS genau ein Attribut *positionLogicalEndOfFile* mit einem ganzzahligen Wert aus dem Intervall [0, *numberOfOctet*] besitzen.**[<=]**

G2_N011.600 - (N011.600) K_Anwendungsspezifikation {K_Karte}
Ein transparentes EF MUSS ein Attribut *body* vom Typ Oktettstring besitzen. Die Anzahl der Oktette in *body* ist gleich *numberOfOctet*. **[<=]**

G2_N011.700 - (N011.700) K_COS
Die Größe einer "data unit" in *body* MUSS 1 Oktett betragen.**[<=]**

G2_N011.800 - (N011.800) K_COS
Für den Offset innerhalb von *body* MUSS gelten: Der Offset des ersten Oktetts in *body* ist null. Ist i der Offset des i–ten Oktetts, dann ist (i + 1) der Offset des nächsten Oktetts.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem Oktettstring zu arbeiten, ist das transparente EF zu selektieren, sofern es nicht bereits *currentEF* ist. Dies geschieht entweder mittels SELECT-Kommando (siehe (N046.700) und (N047.300)), oder mittels *shortFileIdentifier*, der als Parameter dem zugreifenden Kommando mitgegeben wird.

G2_N011.900.a - (N011.900)a K_COS
Zusätzlich zu den in (N011.900)b gelisteten Kommandos MUSS ein transparentes EF auch die folgenden Kommandos unterstützen:
1. ERASE BINARY (siehe (N049.100) und (N049.400)),
2. READ BINARY (siehe (N051.100) und (N051.500)),
3. SET LOGICAL EOF (siehe (N052.932) und (N052.936)),

4. UPDATE BINARY (siehe (N053.200) und (N053.600)).
5. WRITE BINARY (siehe (N055.205) und (N055.226)).**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem Oktettstring zu arbeiten, ist das transparente EF zu selektieren, sofern es nicht bereits *currentEF* ist. Dies geschieht mittels SELECT-Kommando (siehe (N046.700) und (N047.300)).

G2_N011.900.b - (N011.900)b K_COS
Zusätzlich zu den in (N011.900)a gelisteten Kommandos MUSS ein transparentes EF auch die folgenden Kommandos unterstützen:
1. ACTIVATE (siehe (N034.800)),
2. DEACTIVATE (siehe (N036.000)),
3. DELETE (siehe (N037.100)).
4. TERMINATE (siehe (N048.903)).**[<=]**

A_13865 - (N012.000) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für transparente EF weitere Kommandos
a. unterstützt oder
b. ablehnt.**[<=]**

Hinweis CosH_34b: Das Konzept "logical End Of File" unterscheidet zwischen einer Dateigröße, die mittels UPDATE BINARY *beschreibbar ist und einer "logischen" Dateigröße, die mittels* READ BINARY *auslesbar ist. Dies spiegelt sich in den Attributen* numberOfOctet *und* positionLogicalEndOfFile *wieder. Dabei entspricht* numberOfOctet *einer maximalen Dateigröße, die im Rahmen von* UPDATE BINARY *komplett adressierbar ist, und* positionLogicalEndOfFile *einer tatsächlichen Dateigröße, die im Rahmen von* READ BINARY *komplett adressierbar ist. Während der Wert von* numberOfOctet *unveränderlich ist, ist es möglich, dass sich der Wert von* positionLogicalEndOfFile *im Rahmen der Kommandos* SET LOGICAL EOF, UPDATE BINARY *oder* WRITE BINARY *ändert.*

8.3.2.2 Strukturiertes Elementary File

Ein strukturiertes Elementary File (strukturiertes EF) dient der Speicherung einer Liste von Rekords (siehe 8.1.5). Ein Zugriff auf beliebige Listenelemente ist möglich. Der Inhalt des Oktettstrings eines Rekords in strukturierten EF, welche im Rahmen einer Anwendungsspezifikation definiert werden, wird von einem COS lediglich gespeichert, niemals aber interpretiert oder für karteninterne Prozesse verwendet.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein strukturiertes EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13866 - (N012.100) K_Anwendungsspezifikation {K_Karte}
Ein strukturiertes EF ist eine Erweiterung von Datei und MUSS deshalb den Anforderungen aus 8.3.2 genügen.**[<=]**

A_13867 - (N012.200)a K_Anwendungsspezifikation {K_Karte}
Ein strukturiertes EF MUSS ein Attribut *recordList* vom Typ Liste mit einer Anzahl der Listenelemente aus dem Intervall [0, ~~254~~ *maximumNumberOfRecords*] und vom Typ Rekord besitzen.**[<=]**

A_13868 - (N012.200)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für die Anzahl von Listenelementen in *recordList*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N012.300 - (N012.300) K_COS
Das *i*-te Element in *recordList* MUSS die Rekordnummer *i* besitzen.**[<=]**

G2_N012.400.a - (N012.400)a K_Anwendungsspezifikation {K_Karte}
Ein strukturiertes EF MUSS genau ein Attribut *maximumNumberOfRecords* mit einem ganzzahligen Wert aus dem Intervall [1, 254] besitzen.**[<=]**

A_13870 - (N012.400)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *maximumNumberOfRecords*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N012.500.a - (N012.500)a K_Anwendungsspezifikation {K_Karte}
Ein strukturiertes EF MUSS genau ein Attribut *maximumRecordLength* mit einem ganzzahligen Wert aus dem Intervall [1, 255] besitzen.**[<=]**

A_13872 - (N012.500)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *maximumRecordLength*
1. unterstützt oder
2. ablehnt.**[<=]**

A_13873 - (N012.600) K_Anwendungsspezifikation {K_Karte}
Ein strukturiertes EF MUSS genau ein Attribut *flagRecordLifeCycleStatus* vom Typ Boolean besitzen.**[<=]**

G2_N012.600.a - (N012.600)a K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *flagRecordLifeCycleStatus* True ist, dann MÜSSEN alle Elemente in *recordList* einen *lifeCycleStatus* besitzen (siehe (N007.800)).**[<=]**

G2_N012.600.b - (N012.600)b K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *flagRecordLifeCycleStatus* False ist, dann DARF KEIN Element in *recordList* einen *lifeCycleStatus* besitzen.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem Oktettstring zu arbeiten, ist das strukturierte EF zu selektieren, sofern es nicht bereits *currentEF* ist. Dies geschieht entweder mittels SELECT-Kommando (siehe (N046.700) und (N047.300)), oder mittels *shortFileIdentifier*, der als Parameter dem zugreifenden Kommando mitgegeben wird.

G2_N012.700.a - (N012.700)a K_COS
Zusätzlich zu den in (N012.700)b gelisteten Kommandos MUSS ein sturkturiertes EF auch die folgenden Kommandos unterstützen:
1. ACTIVATE RECORD (siehe (N055.500), (N055.900), (N056.200) und (N056.600)),
2. APPEND RECORD (siehe (N058.400) und (N058.700)),
3. DEACTIVATE RECORD (siehe (N060.700), (N061.100), (N061.400) und (N061.800)),
4. DELETE RECORD (siehe (N063.422) und (N063.426)),
5. ERASE RECORD (siehe (N063.600) und (N063.900)),
6. READ RECORD (siehe (N065.700) und (N066.100)),
7. SEARCH RECORD (siehe (N067.900) und (N068.400)),
8. UPDATE RECORD (siehe (N070.300) und (N070.700)).**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist spezifikationsgemäß zu arbeiten, ist das strukturierte EF zu selektieren, sofern es nicht bereits *currentEF* ist. Dies geschieht mittels SELECT-Kommando (siehe (N046.700) und (N047.300)).

G2_N012.700.b - (N012.700)b K_COS

Zusätzlich zu den in (N012.700)a gelisteten Kommandos MUSS ein strukturiertes EF auch die folgenden Kommandos unterstützen:
1. ACTIVATE (siehe (N034.800)),
2. DEACTIVATE (siehe (N036.000)),
3. DELETE (siehe (N037.100)).
4. TERMINATE (siehe (N048.903)).**[<=]**

A_13874 - (N012.800) K_TST
~~Ein strukturiertes EF KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Kommandos
a. unterstützt oder
b. ablehnt.**[<=]**

8.3.2.2.1 Linear variables Elementary File

Ein linear variables Elementary File (linear variables EF) dient der Speicherung einer Liste von Elementen des Typs *record* (siehe 8.1.5), wobei es möglich ist, dass der Oktettstring eines jeden Listenelementes eine andere Anzahl von Oktette enthält.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein linear variables EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13892 - (N012.900) K_Anwendungsspezifikation {K_Karte}
Ein linear variables EF ist eine Erweiterung des strukturierten EF und MUSS deshalb den Anforderungen aus 8.3.2.2 genügen.**[<=]**

G2_N013.000.a - (N013.000)a K_Anwendungsspezifikation {K_Karte}
Ein linear variables EF MUSS genau ein Attribut *numberOfOctet* mit einem ganzzahligen Wert aus dem Intervall [1, 64.770] besitzen.**[<=]**

A_13894 - (N013.000)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *numberOfOctet*
1. unterstützt oder
2. ablehnt.**[<=]**

A_13895 - (N013.050) K_Anwendungsspezifikation {K_Karte}
KEIN *record* in *recordList* DARF mehr als *maximumRecordLength* Oktette besitzen.**[<=]**

A_17463 - (N013.060) K_Anwendungsspezifikation {K_Karte}
Die Summe der Längen in Oktett aller Elemente in *recordList* MUSS kleiner gleich *numberOfOctet* sein.**[<=]**

G2_N013.100 - (N013.100) K_COS
Wenn mittels APPEND RECORD Kommando der Liste *recordList* ein neuer *record* hinzugefügt, dann MUSS das COS den neuen *record* am Ende der Liste einfügen.**[<=]**

8.3.2.2.2 Linear fixes Elementary File

Ein linear fixes Elementary File (linear fixes EF) dient der Speicherung einer Liste von Elementen des Typs *record* (siehe 8.1.5), wobei der Oktettstring eines jeden Listenelementes dieselbe Anzahl von Oktette enthält.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein linear fixes EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13896 - (N013.200) K_Anwendungsspezifikation {K_Karte}
Ein linear fixes EF ist eine Erweiterung des strukturierten EF und MUSS deshalb den Anforderungen aus 8.3.2.2 genügen.**[<=]**

G2_N013.300 - (N013.300) K_Anwendungsspezifikation {K_Karte}
Jeder *record* in *recordList* MUSS *maximumRecordLength* Oktette besitzen.**[<=]**

G2_N013.400 - (N013.400) K_COS
Wenn mittels APPEND RECORD Kommando der Liste *recordList* ein neuer *record* hinzugefügt, dann MUSS das COS den neuen *record* am Ende der Liste eingefügen.**[<=]**

G2_N013.410 - (N013.410) K_Anwendungsspezifikation {K_Karte}
Das Produkt aus *maximumNumberOfRecords* und *maximumRecordLength* MUSS kleiner gleich 64.770 sein (vergleiche auch (N013.000)a).**[<=]**

8.3.2.2.3 Zyklisches Elementary File

Ein zyklisches Elementary File (zyklisches EF) dient der Speicherung einer Liste von Elementen des Typs *record* (siehe 8.1.5), wobei der Oktettstring eines jeden Listenelementes dieselbe Anzahl von Oktette enthält.

Gemäß der Norm [ISO/IEC 7816-4] gelten für ein zyklisches EF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

A_13897 - (N013.500) K_Anwendungsspezifikation {K_Karte}
Ein zyklisches EF ist eine Erweiterung des strukturierten EF und MUSS deshalb den Anforderungen aus 8.3.2.2 genügen.**[<=]**

G2_N013.600 - (N013.600) K_Anwendungsspezifikation {K_Karte}
Jeder *record* in *recordList* MUSS *maximumRecordLength* Oktette besitzen.**[<=]**

G2_N013.700 - (N013.700) K_COS
Wenn mittels APPEND RECORD Kommando der Liste *recordList* ein neuer *record* hinzugefügt, dann MUSS das COS den neuen *record* am Anfang der Liste eingefügen.**[<=]**

G2_N013.800 - (N013.800) K_Anwendungsspezifikation {K_Karte}
Zyklische Elementary Files MÜSSEN für das Attribut *flagRecordLifeCycleStatus* den Wert False verwenden.**[<=]**

G2_N013.810 - (N013.810) K_Anwendungsspezifikation {K_Karte}
Das Produkt aus *maximumNumberOfRecords* und *maximumRecordLength* MUSS kleiner gleich 64.770 sein (vergleiche auch (N013.000)a).**[<=]**

8.3.3 File Control Parameter

Die File Control Parameter enthalten Attribute eines Files. Im Rahmen dieses Dokumentes werden sie lediglich an der Schnittstelle Interpreter (siehe CosA_e09) in den Antwortdaten eines SELECT-Kommandos (siehe (N048.300)a) sichtbar. Für die Codierung der File Control Parameter gilt:

G2_N013.900 - (N013.900) K_COS
Das COS MUSS die File Control Parameter (FCP) gemäß DER-TLV in einem DO_FCP mit Tag = '62' codieren.**[<=]**

A_13909 - (N014.000) K_COS

DO'80': Wenn das File vom Typ transparentes EF (siehe8.3.2.1) oder linear variables EF (siehe 8.3.2.2.1) ist, genau dann MUSS das COS in DO_FCP ein DO_Size einstellen.**[<=]**

G2_N014.000.a - (N014.000)a K_COS
Das COS MUSS für DO_Size ein Tag = '80' verwenden.**[<=]**

A_13910 - (N014.000)b K_externeWelt {K_Karte}
~~Das Wertfeld KANN eine beliebige Länge besitzen.~~ Die externe Welt MUSS beliebige Längen des DO_Size akzeptieren. **[<=]**

G2_N014.000.c - (N014.000)c K_COS
Das COS MUSS das Wertfeld von DO_Size so codieren, dass das Ergebnis einer Konvertierung mittels OS2I(…) gleich *numberOfOctet* (siehe (N011.500)a, bzw. (N013.000)a) der Datei ist.**[<=]**

A_13911 - (N014.100) K_COS
DO'82': Das COS MUSS in DO_FCP ein DO_FileDescriptor einstellen.**[<=]**

G2_N014.100.a - (N014.100)a K_COS
Wenn das File vom Typ Ordner (siehe 8.3.1) ist, dann MUSS das COS DO_FileDescriptor wie folgt codieren:
1. DO_FileDescriptor MUSS ein Tag = '82' besitzen.
2. Das Wertfeld MUSS aus einem Oktett bestehen.
3. Das Wertfeld MUSS einen Wert aus der Menge {'38', '78'} enthalten.**[<=]**

G2_N014.100.b - (N014.100)b K_COS
Wenn das File vom Typ transparentes EF (siehe 8.3.2.1) ist, dann MUSS das COS DO_FileDescriptor wie folgt codieren:
1. DO_FileDescriptor MUSS ein Tag = '82' besitzen.
2. Das Wertfeld MUSS aus einem Oktett bestehen.
3. Das Wertfeld MUSS einen Wert aus der Menge {'01', '41'} enthalten.**[<=]**

G2_N014.100.c - (N014.100)c K_COS
Wenn das File vom Typ strukturiertes EF (siehe 8.3.2.2) ist, dann MUSS das COS DO_FileDescriptor wie folgt codieren:
1. DO_FileDescriptor MUSS ein Tag = '82' besitzen.
2. Die Länge des Wertfeldes von DO_FileDescriptor MUSS aus der Menge {5, 6} sein.
3. Das erste Oktett des Wertfeld MUSS einen Wert aus der Menge
   i. {'02', '42'} besitzen, wenn File vom Typ linear fixes EF ist.
   ii. {'04', '44'} besitzen, wenn File vom Typ linear variables EF ist.
   iii. {'06', '46'} besitzen, wenn File vom Typ zyklisches EF ist.
4. Das zweite Oktett im Wertfeld MUSS den Wert '41' besitzen.
5. Das 3. und 4. Oktett MUSS gleich I2OS(*maximumRecordLength*, 2) sein.
6. Das fünfte Oktett MUSS bzw. das fünfte und sechste Oktett MÜSSEN so gewählt werden, dass deren OS2I Konvertierung das Attribut *maximumNumberOfRecords* liefert.**[<=]**

G2_N014.200 - (N014.200) K_COS
DO'83': Wenn das File ein Attribut vom Typ*fileIdentifier* gemäß 8.1.1 besitzt, genau dann MUSS das COS in DO_FCP ein DO_FID einstellen, wobei gilt:
DO_FID = '83 02 || I2OS(*fileIdentifier*, 2)'.**[<=]**

G2_N014.300 - (N014.300) K_COS
DO'84': Wenn das File Attribute vom Typ*applicationIdentifier* gemäß (N010.200) besitzt, dann MUSS das COS in DO_FCP jedes dieser Attribute einstellen. Jedes dieser Attribute

wird als Wertfeld in einem DO_AID codiert und es gilt:
DO_AID = '84 || I2OS(OctetLength(*applicationIdentifier*, 1) || *applicationIdentifier*'.**[<=]**

A_13912 - (N014.400) K_COS
DO'88': Wenn das File vom Typ Datei (siehe8.3.2) ist, genau dann MUSS das COS in DO_FCP ein DO_SFI einstellen.**[<=]**

G2_N014.400.a - (N014.400)a K_COS
Wenn die Datei ein Attribut *shortFileIdentifier* gemäß (N007.000) besitzt, dann MUSS das COS DO_SFI wie folgt codieren:
DO_SFI = '88 01 || I2OS(8*shortFileIdentifier*, 1)'.**[<=]**

G2_N014.400.b - (N014.400)b K_COS
Wenn die Datei kein Attribut *shortFileIdentifier* gemäß (N007.000) besitzt, dann MUSS das COS DO_SFI wie folgt codieren:
DO_SFI = '88 00'.**[<=]**

A_13913 - (N014.500) K_COS
DO'8A': Das COS MUSS den physikalischen Wert des Attributes*lifeCycleStatus* (siehe (N007.100)a) eines Files in DO_FCP als DO_LCS mit einem Tag = '8A' und einem Wertfeld der Länge eins einstellen. Der Wert von*lifeCycleStatus*wird gemäß [ISO/IEC 7816-4#Tabelle 14] codiert.**[<=]**

G2_N014.500.a - (N014.500)a K_COS
Das COS MUSS für *lifeCycleStatus* == "Operational state (active)" einen Wert aus der Menge {'05', '07'} verwenden.**[<=]**

G2_N014.500.b - (N014.500)b K_COS
Das COS MUSS für *lifeCycleStatus* == "Operational state (deactivated)" einen Wert aus der Menge {'04', '06'} verwenden.**[<=]**

G2_N014.500.c - (N014.500)c K_COS
Das COS MUSS für *lifeCycleStatus* == "Termination state" einen Wert aus der Menge {'0C'. '0D', '0E', '0F'} verwenden.**[<=]**

G2_N014.600.a - (N014.600)a K_COS
DO'8F': Wenn das File vom Typ strukturiertes EF (siehe8.3.2.2) ist und das Attribut *flagRecordLifeCycleStatus* hat den Wert False, dann MUSS das COS
1. entweder in den DO_FCP ein DO_Profile = '8F 01 00' einstellen,
2. oder auf ein DO mit Tag = '8F' in den DO_FCP verzichten.**[<=]**

G2_N014.600.b - (N014.600)b K_COS
DO'8F': Wenn das File vom Typ strukturiertes EF (siehe8.3.2.2) ist und das Attribut *flagRecordLifeCycleStatus* hat den Wert True, genau dann MUSS das COS in DO_FCP ein DO_ProfileIdentifier = '8F 01 XX' einstellen, wobei das Wertfeld dieses DOs gemäß OS2I('XX') convertiert eine ungerade Zahl kleiner als 128 ist.**[<=]**

A_13914 - (N014.700) K_COS
DO'C5': Wenn das File vom Typ transparentes EF (siehe8.3.2.1) ist, genau dann MUSS das COS in DO_FCP ein DO_ReadSize einstellen.**[<=]**

G2_N014.700.a - (N014.700)a K_COS
Das COS MUSS für DO_ReadSize ein Tag = 'C5' verwenden.**[<=]**

G2_N014.700.b - (N014.700)b K_externeWelt {K_Karte}
~~Das Wertfeld KANN eine beliebige Länge besitzen.~~ Die externe Welt MUSS beliebige Längen des DO_ReadSize akzeptieren. **[<=]**

G2_N014.700.c - (N014.700)c K_COS
Das COS MUSS das Wertfeld von DO_Size so codieren, dass das Ergebnis einer Konvertierung mittels OS2I(…) gleich *positionLogicalEndOfFile* (siehe (N011.510)) der Datei ist.**[<=]**

A_13915 - (N014.800) K_TST
~~In DO_FCP KÖNNEN~~ Für die funktionale Eignung MUSS es zulässig sein, dass DO_FCP weitere Datenobjekte enthält. Enthält DO_FCP weitere Datenobjekte, so ist deren Codierung und Bedeutung herstellerspezifisch.**[<=]**

A_13916 - (N014.900)a K_externeWelt {K_Karte}
~~Für die Reihenfolge der Datenobjekte in DO_FCP gilt: Die Reihenfolge aller Datenobjekte ist herstellerspezifisch.~~ Die externe Welt MUSS jede beliebige Reihenfolge der DO in DO_FCP akzeptieren. **[<=]**

G2_N014.900.b - (N014.900)b K_COS
Wenn in DO_FCP vorhanden, dann MUSS das COS die Reihenfolge der DO in DO_FCP so wählen, dass die Datenobjekte DO_Size, DO_FileDescriptor, DO_FID, DO_AID, DO_SFI, DO_LCS, DO_ProfileIdentifier und DO_ReadSize vollständig in den ersten 256 Oktetten des Oktettstrings DO_FCP enthalten sind.**[<=]**

8.4 Reguläres-Passwort (normativ)

Hinweis CosH_f6b: Mit der Version 3.0 dieses Dokumentes wurde der Objekttyp "Multireferenz-Passwort" eingeführt (siehe8.5). An der Schnittstelle ist es nicht möglich zwischen einem "Regulären-Passwort", welches alle Attribute selbst besitzt und einem "Multireferenz-Passwort", welches gewisse Attribute eines "Regulären-Passworts" nachnutzt, zu unterscheiden. Diesbezüglich ist die Unterscheidung in diesem Dokument willkürlich und einerseits darin begründet die Unterschiede zu älteren Versionen dieses Dokumentes nicht unnötig zu vergrößern und andererseits eine einfache und präzise Darstellung zu ermöglichen. Als Konsequenz werden die Begriffe "Reguläres-Passwort" und "Multireferenz-Passwort" nur verwendet, wenn die Unterscheidung wichtig ist. An Stellen, wo diese Unterscheidung nicht wichtig ist wird der Begriff "Passwort" verwendet.

Ein "Reguläres-Passwort" dient der Speicherung eines Geheimnisses, das in der Regel nur einem Karteninhaber bekannt ist. Das COS wird bestimmte Dienste erst dann zulassen, wenn dieses Geheimnis im Rahmen einer Benutzerverifikation erfolgreich präsentiert wurde. Die Notwendigkeit der Benutzerverifikation lässt sich einschalten (enable) oder abschalten (disable).

Typischerweise ist das Geheimnis aus Sicherheitsgründen änderbar. Ebenfalls aus Sicherheitsgründen wird die Operation "Benutzerverifikation" für ein bestimmtes Passwort vom COS gesperrt, wenn die Benutzerverifikation mit diesem Passwort zu oft fehlschlug. Typischerweise ist es möglich, diese Sperrung aufzuheben.

Hinweis CosH_b0a: In anderen Dokumenten wird in der Regel der Begriff PIN verwendet und bezeichnet dann teilweise das einem Karteninhaber bekannte sechsstellige Geheimnis ("Bitte geben Sie Ihre PIN ein.") und teilweise das Objekt in seiner Gesamtheit ("Die PIN ist blockiert."). In diesem Dokument wird der sprachlichen Klarheit wegen, weitgehend auf den Begriff PIN verzichtet und zur Bezeichnung des Objekttyps stets der Begriff Passwort verwendet.

Bei der Spezifikation von Applikationen sind folgende Regeln einzuhalten:

G2_N015.000.a - (N015.000)a K_Anwendungsspezifikation {K_Karte}

Ein Reguläres-Passwort MUSS genau ein Attribut *pwdIdentifier* mit einem ganzzahligen Wert aus dem Intervall [0, 31] besitzen.**[<=]**

A_13923 - (N015.000)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *pwdIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N015.050 - (N015.050) K_Anwendungsspezifikation {K_Karte}
Eine Reguläres-Passwort MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100)a) besitzen.**[<=]**

G2_N015.100 - (N015.100) K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

G2_N015.200 - (N015.200) K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *secret* vom Typ *pin* (siehe 8.1.7) besitzen.**[<=]**

G2_N015.300.a - (N015.300)a K_Anwendungsspezifikation {K_Karte}
Das Reguläres-Passwort MUSS genau ein Attribut *minimumLength* mit einem ganzzahligen Wert aus dem Intervall [4, 12] besitzen.**[<=]**

A_13925 - (N015.300)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *minimumLength*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N015.310.a - (N015.310)a K_Anwendungsspezifikation {K_Karte}
Das Reguläres-Passwort MUSS genau ein Attribut *maximumLength* mit einem ganzzahligen Wert aus dem Intervall [*minimumLength*, 12] besitzen.**[<=]**

A_13927 - (N015.310)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *maximumLength*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N015.400.a - (N015.400)a K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *startRetryCounter* mit einem ganzzahligen Wert aus dem Intervall [1, 15] besitzen.**[<=]**

A_13929 - (N015.400)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *startRetryCounter*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N015.500.a - (N015.500)a K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *retryCounter* mit einem ganzzahligen Wert aus dem Intervall [0, 15] besitzen.**[<=]**

A_13931 - (N015.500)b K_TST

~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *retryCounter*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N015.600 - (N015.600) K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *transportStatus* gemäß (N009.500) besitzen.**[<=]**

G2_N015.610 - (N015.610) K_COS {K_Karte}
Wenn das Attribut *transportStatus* eines regulären Passwortobjektes vom Zustand "Leer-PIN" in den Zustand "regularPassword" wechselt, dann MUSS das COS auf eine herstellerspezifische Art und Weise sicherstellen, dass der Use Case "Setzen eines Benutzergeheimnisses" gemäß (N073.700) für dieses reguläre Passwortobjekt und alle auf dieses reguläre Passwortobjekt verweisenden Multireferenz-Passwortobjekte nicht mehr ausführbar ist.**[<=]**

G2_N015.700 - (N015.700) K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *flagEnabled* vom Typ Boolean besitzen. Es wird im Rahmen der Zugriffsregelauswertung verwendet (siehe (N022.200)a.2).**[<=]**

A_13932 - (N015.800) K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *startSsecList* besitzen.**[<=]**

G2_N015.800.a - (N015.800)a K_Anwendungsspezifikation {K_Karte}
Die Liste *startSsecList* MUSS eine Länge aus dem Intervall [1, 4] haben.**[<=]**

A_13933 - (N015.800)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Listenlängen für *startSsecList*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N015.800.c - (N015.800)c K_Anwendungsspezifikation {K_Karte}
Jedes Listenelement in *startSsecList* MUSS
1. genau einen *seIdentifier* gemäß (N007.900) und
2. genau eine ganze Zahl *startSsec* enthalten, deren Wert entweder aus dem Intervall [1, 250] zu wählen ist oder die den Wert "unendlich" repräsentiert.**[<=]**

A_13934 - (N015.800)d K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *startSsec* würden
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N015.900 - (N015.900) K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS genau ein Attribut *PUK* vom Typ *pin* (siehe 8.1.7) besitzen, welches ein Geheimnis repräsentiert, welches das Zurücksetzten eines abgelaufenen *retryCounter* ermöglicht.**[<=]**

G2_N016.000.a - (N016.000)a K_Anwendungsspezifikation {K_Karte}
Ein Reguläres-Passwort MUSS ein Attribut *pukUsage* mit einem ganzzahligen Wert aus dem Intervall [0, 15] besitzen.**[<=]**

A_13936 - (N016.000)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *pukUsage*

1. unterstützt oder
2. ablehnt.**[<=]**

G2_N016.100.a - (N016.100)a K_externeWelt {K_Karte}
Die Ziffernfolgen *secret* und *PUK* MÜSSEN an der Schnittstelle Interpreter (siehe CosA_e09) stets als Format-2-PIN-Block (siehe (N008.100)) übertragen werden.**[<=]**

A_13937 - (N016.100)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Übertragungsformate
1. unterstützt oder
2. ablehnt.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem Passwort zu arbeiten, ist das Passwort zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.

A_13938 - (N016.200) K_COS
Das COS MUSS für den Objekttyp Passwort die folgenden Kommandos unterstützen:
a. reguläre Nutzung:
   1. CHANGE REFERENCE DATA (siehe (N073.300) und (N073.700)),
   2. DISABLE VERIFICATION REQUIREMENT (siehe (N075.386) und (N075.500)),
   3. ENABLE VERIFICATION REQUIREMENT (siehe (N076.586) und (N076.700)),
   4. GET PIN STATUS (siehe (N077.900)),
   5. RESET RETRY COUNTER (siehe (N079.300) und (N079.700) und (N080.100) und (N080.300)),
   6. VERIFY (siehe (N082.200)).
b. administrative Kommandos:
   1. ACTIVATE (siehe (N034.834)),
   2. DEACTIVATE (siehe (N036.034)),
   3. DELETE (siehe (N037.134)),
   4. TERMINATE (siehe (N048.934)).**[<=]**

A_13939 - (N016.210) K_externeWelt {K_Karte}
Die externe Welt DARF ein reguläres Passwort NICHT löschen (Use Case entsprechend (N037.134)), solange es Multireferenz-Passwortobjekte gibt, die mittels ihres Attributes *passwordReference* (siehe (N016.320)f) auf dieses reguläre Passwortobjekt verweisen.
[<=]

A_13940 - (N016.300) K_TST
~~Ein Reguläres-Passwort KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für Reguläre-Passworte weitere Kommandos
a. unterstützt oder
b. ablehnt.**[<=]**

Einem Regulären-Passwort ist ein weiteres, kanalspezifisches Attribut *securityStatusEvaluationCounter* (siehe (N029.900)k) zugeordnet, das typischerweise einem flüchtig (etwa im RAM) gespeicherten Sicherheitszustand (siehe 8.9) zugerechnet wird. Es wird im Rahmen von Kommandos zur Benutzerverifikation (siehe (N082.200)) geändert.

Hinweis CosH_1e8: Absichtlich ist einem Passwort kein Attribut shareable *zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein Passwortobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.*

8.5 Multireferenz-Passwort (normativ)

Ein Multireferenz-Passwort ermöglicht die Nutzung eines Geheimnisses, welches als Attribut in einem Regulären-Passwort gespeichert ist (siehe (N015.200)), allerdings unter Bedingungen, die von denen des Regulären-Passwortes bewusst abweichen. Wird das Verhalten an der Schnittstelle zur Smartcard betrachtet, so scheinen Reguläres-Passwort und Multireferenz-Passwort gewisse Attribute gemeinsam zu nutzen, während es für die übrigen Attribute möglich ist diese individuell verschieden zu wählen. Aus Sicht eines Karteninhabers scheinen die im bekannten Geheimnisse (*secret* gemäß (N015.200) und *PUK* gemäß (N015.900)) unter mehreren "Referenzen" (genauer*pwdIdentifiern* gemäß (N015.000)) ansprechbar zu sein.

Bei der Spezifikation von Applikationen sind folgende Regeln einzuhalten:

A_13951 - (N016.320) K_Anwendungsspezifikation {K_Karte}
Folgende Attribute MÜSSEN für ein Multireferenz-Passwort individuell festlegt werden, das heißt diese Attribute werden nicht gemeinsam mit anderen Regulären-Passwörtern oder anderen Multireferenz-Passwörtern genutzt:**[<=]**

G2_N016.320.a - (N016.320)a K_Anwendungsspezifikation {K_Karte}
pwdIdentifier: Ein Multireferenz-Passwort MUSS genau ein Attribut *pwdIdentifier* besitzen. Für dieses Attribut gelten die in (N015.000) genannten Anforderungen.**[<=]**

G2_N016.320.b - (N016.320)b K_Anwendungsspezifikation {K_Karte}
lifeCycleStatus: Eine Multireferenz-Passwort MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100)a) besitzen.**[<=]**

G2_N016.320.c - (N016.320)c K_Anwendungsspezifikation {K_Karte}
accessRules: Ein Multireferenz-Passwort MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

G2_N016.320.d - (N016.320)d K_Anwendungsspezifikation {K_Karte}
startSsecList: Ein Multireferenz-Passwort MUSS genau ein Attribut *startSsecList* besitzen. Für diese Liste gelten die in (N015.800) genannten Anforderungen.**[<=]**

G2_N016.320.e - (N016.320)e K_Anwendungsspezifikation {K_Karte}
flagEnabled: Ein Multireferenz-Passwort MUSS genau ein Attribut *flagEnabled* besitzen. Für dieses Attribut gelten die in (N015.700) genannten Anforderungen.**[<=]**

G2_N016.320.f - (N016.320)f K_Anwendungsspezifikation {K_Karte}
passwordReference: Ein Multireferenz-Passwort MUSS genau ein Attribut *passwordReference* mit den folgenden Eigenschaften besitzen.
1. Das Attribute *passwordReference* enthält eine Referenz auf ein Passwort.
2. Sei *objectFolder* der Ordner, welcher dieses Multireferenz-Passwort enthält, dann gilt:
   i. Die Funktion *pwd* = SearchPwd(*objectFolder*, *passwordReference*) meldet nicht den Fehler *pwdNotFound* zurück.
   ii. *pwd* ist ein Reguläres-Passwort (keine Kettenbildung).
   iii. Die Attribute, welche dieses Multireferenz-Passwort gemeinsam mit einem Regulären-Passwort nutzt, werden dem Passwort *pwd* entnommen. Mit anderen Worten: Die Attribute, welches das Multireferenz-Passwort nicht selber speichert, steuert *pwd* bei.
[<=]

Folgende Attribute werden von einem Mutlireferenz-Passwort und dem mittels des Attributes *passwordReference* referenzierten Regulären-Passwort *pwd* gemeinsam genutzt:

G2_N016.325.a - (N016.325)a K_COS

Das COS MUSS das Attribute *secret* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

G2_N016.325.b - (N016.325)b K_COS
Das COS MUSS das Attribute *minimumLength* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

G2_N016.325.c - (N016.325)c K_COS
Das COS MUSS das Attribute *maximumLength* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

G2_N016.325.d - (N016.325)d K_COS
Das COS MUSS das Attribute *startRetryCounter* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

G2_N016.325.e - (N016.325).e K_COS
Das COS MUSS das Attribute *retryCounter* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

G2_N016.325.f - (N016.325)f K_COS
Das COS MUSS das Attribute *transportStatus* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

G2_N016.325.g - (N016.325)g K_COS
Das COS MUSS das Attribute *PUK* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

G2_N016.325.h - (N016.325)h K_COS
Das COS MUSS das Attribute *pukUsage* gemeinsam mit dem gleichnamigen aus *pwd* nutzen.**[<=]**

A_13952 - (N016.330) K_COS
Ein Multireferenz-Passwort MUSS dieselben Kommandos unterstützen wie für ein Reguläres-Passwort in (N016.200) gefordert.**[<=]**

A_13953 - (N016.335) K_TST
~~Ein Multireferenz-Passwort KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für Multireferenz-Passworte weitere Kommandos
a. unterstützt oder
b. ablehnt.**[<=]**

Einem Multireferenz-Passwort ist ein weiteres, kanalspezifisches Attribut *securityStatusEvaluationCounter* (siehe (N029.900)k) zugeordnet, das typischerweise einem flüchtig (etwa im RAM) gespeicherten Sicherheitszustand (siehe 8.9) zugerechnet wird. Es wird im Rahmen von Kommandos zur Benutzerverifikation (siehe (N082.200)) geändert.

8.6 Schlüsselobjekt (normativ)

Dieses Unterkapitel beschreibt Schlüsselobjekte, die im Rahmen kryptographischer Operationen zum Einsatz kommen. Der Terminus Schlüsselobjekt dient in diesem Dokument als Oberbegriff für symmetrische, private und öffentliche Schlüsselobjekte.

Symmetrische Schlüssel werden in diesem Dokument zu folgenden Zwecken eingesetzt:

~~1. Mit persistent gespeichertem Geheimnis (Schlüssel) zur einseitigen Authentisierung (siehe 15.1.1).~~

1. Mit persistent gespeichertem Geheimnis (Schlüssel) zur gegenseitigen Authentisierung bei gleichzeitiger Aushandlung von Sessionkeys (siehe 15.4.1 und 15.4.2).
2. Mit persistent gespeichertem Geheimnis (Schlüssel) zur gegenseitigen Authentisierung bei gleichzeitiger Übertragung von Sessionkeys (siehe 15.5).
3. Als Sessionkey zur Sicherstellung einer vertraulichen Kommunikation.
4. Als Sessionkey zur Sicherstellung einer integren und authentischen Kommunikation.

Private Schlüssel werden in diesem Dokument zu folgenden Zwecken eingesetzt:

5. Berechnung elektronischer Signaturen (siehe 14.8.2)
6. Entschlüsselung von Daten (siehe 14.8.3)
7. Nachweis der Authentizität dieser Karte (siehe 15.2)
8. Transportsicherung von Sessionkey Material (siehe 15.5 und (N085.068)b.7.viii~~(N084.400)c~~)

Öffentliche Schlüssel werden in diesem Dokument zu folgenden Zwecken eingesetzt:

9. Prüfen elektronischer Signaturen beim Import von Zertifikaten (siehe (N095.900))
10. Prüfen von Signaturen im Rahmen von Rollenauthentisierungen (siehe (N084.400))
11. Transportsicherung von Sessionkey Material (siehe 15.5 und (N085.068)b.7.viii~~(N086.900)d~~)
12. Verschlüsseln von Daten (siehe 14.8.4)

8.6.1 Symmetrisches Authentisierungsobjekt

Ein symmetrisches Authentisierungsobjekt wird im Rahmen von Authentisierungen gemäß 15.4.1 und 15.5 eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N016.400.a - (N016.400)a K_Anwendungsspezifikation {K_Karte}
Ein symmetrisches Authentisierungsobjekt MUSS ein Attribut *keyIdentifier* mit einem ganzzahligen Wert aus dem Intervall [2, 28] besitzen.**[<=]**

A_13958 - (N016.400)b K_TST
~~En COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *keyIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N016.450 - (N016.450) K_Anwendungsspezifikation {K_Karte}
Eine symmetrisches Authentisierungsobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100)a) besitzen.**[<=]**

G2_N016.500 - (N016.500) K_Anwendungsspezifikation {K_Karte}

Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

G2_N016.590.a - (N016.590)a K_Anwendungsspezifikation {K_Karte}
Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *keyType* aus der Menge {~~3TDES (Option_DES),~~ AES-128, AES-192, AES-256} besitzen.**[<=]**

A_13959 - (N016.590)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *keyType*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N016.600 - (N016.600) K_Anwendungsspezifikation {K_Karte}
Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *encKey* gemäß 8.2.1 besitzen.**[<=]**

G2_N016.700 - (N016.700) K_Anwendungsspezifikation {K_Karte}
Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *macKey* gemäß 8.2.1 besitzen.**[<=]**

G2_N016.705 - (N016.705) K_Anwendungsspezifikation {K_Karte}
Die Attribute *encKey* und *macKey* MÜSSEN passend zum Attribut *keyType* gewählt werden.**[<=]**

G2_N016.710 - (N016.710) K_Anwendungsspezifikation {K_Karte}
Wenn *algorithmIdentifier* Element der Menge {aesSessionkey4SM} ist, genau dann MUSS ein symmetrisches Authentisierungsobjekt ein Attribut *numberScenario* vom Typ ganze Zahl aus dem Intervall [0, 32.767] besitzen.**[<=]**

A_13960 - (N016.800) K_Anwendungsspezifikation {K_Karte}
Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut vom Typ *algorithmIdentifier* besitzen, welches angibt, für welchen Zweck es verwendbar ist.**[<=]**

G2_N016.800.a - (N016.800)a K_Anwendungsspezifikation {K_Karte}
Für symmetrische Authentisierungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
1. aesSessionkey4SM, (siehe, (N084.410)a)
~~2. desSessionkey4SM (siehe, (N084.410)b, Option_DES)~~
} sein (siehe CosT_2a4).**[<=]**

G2_N016.800.b - (N016.800)b K_Anwendungsspezifikation {K_Karte}, Option_Kryptobox
Für symmetrische Authentisierungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
1. aesSessionkey4TC, (siehe (N084.402)a, (N086.902)a),
2. aesSessionkey4SM, (siehe, (N084.410)a)
~~2. desSessionkey4TC (siehe (N084.402)b, (N086.902)b, Option_DES)~~
} sein (siehe CosT_2a4).**[<=]**

A_13961 - (N016.800)c K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algorithmIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N016.820 - (N016.820) K_Anwendungsspezifikation {K_Karte}, Option_Kryptobox
Wenn *algorithmIdentifier* Element der Menge {aesSessionkey4TC~~, desSessionkey4TC~~} ist, genau dann MUSS ein symmetrisches Authentisierungsobjekt ein Attribut *accessRulesSessionkeys* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Authentisierungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels der in 14.9.9 beschriebenen Use Cases.

G2_N016.900.a - (N016.900)a K_COS
Zusätzlich zu den in (N016.900)b gelisteten Kommandos MUSS ein symmetrischen Authentisierungsobjekt auch die folgenden Kommandos unterstützen:
1. EXTERNAL AUTHENTICATE (siehe (N083.500) und (N101.400)),
2. GENERAL AUTHENTICATE (siehe (N085.022) und (N101.400)),
3. INTERNAL AUTHENTICATE (siehe (N086.400) und (N100.400)),
4. MUTUAL AUTHENTICATE (siehe (N083.800) und (N102.400)).**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Authentisierungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.

G2_N016.900.b - (N016.900)b K_COS
Zusätzlich zu den in (N016.900)a gelisteten Kommandos MUSS ein symmetrischen Authentisierungsobjekt auch die folgenden Kommandos unterstützen:
1. ACTIVATE (siehe (N034.814)),
2. DEACTIVATE (siehe (N036.014)),
3. DELETE (siehe (N037.144)).
~~4. GET SECURITY STATUS KEY (siehe (N085.200), Option_DES),~~
4. TERMINATE (siehe (N048.914)).**[<=]**

A_13962 - (N017.000) K_TST
~~Ein symmetrisches Authentisierungsobjekt KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für symmetrische Authentisierungsobjekte weitere Kommandos
1. unterstützt oder
2. ablehnt.**[<=]**

Hinweis CosH_ef7: Absichtlich ist einem symmetrischen Authentisierungsobjekt kein Attribut shareable *zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein symmetrisches Authentisierungsobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.*

8.6.2 Symmetrisches Kartenverbindungsobjekt

Ein symmetrisches Kartenverbindungsobjekt wird im Rahmen von Authentisierungen gemäß 15.4.2 verwendet. Die normativen Festlegungen in diesem Dokument bilden eine Untermenge der in [BSI-TR-03110-3] für eine CAN (Card Access Number) beschriebenen Funktionalität: Ein symmetrisches Kartenverbindungsobjekt ist nicht änderbar und unterstützt ausschließlich PACE in der Version 2 in der ECDH-Variante.

Für symmetrische Kartenverbindungsobjekte gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N017.020.a - (N017.020)a K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Ein symmetrisches Kartenverbindungsobjekt MUSS ein Attribut *keyIdentifier* mit einem ganzzahligen Wert aus dem Intervall [2, 28] besitzen.**[<=]**

A_13966 - (N017.020)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *keyIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N017.024 - (N017.024) K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Eine symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007,100)a) besitzen.**[<=]**

G2_N017.026 - (N017.026) K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

G2_N017.028 - (N017.028) K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut *can* vom Typ *pin* gemäß (N008.000) besitzen.**[<=]**

A_13967 - (N017.030) K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut vom Typ *algorithmIdentifier* besitzen, welches angibt, für welchen Zweck es verwendbar ist (siehe 15.4.2). Folgende Werte sind zulässig:**[<=]**

G2_N017.030.a.1 - (N017.030)a.1 K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle
algorithmIdentifier = id-PACE-ECDH-GM-AES-CBC-CMAC-128,**[<=]**

G2_N017.030.a.2 - (N017.030)a.2 K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle
algorithmIdentifier = id-PACE-ECDH-GM-AES-CBC-CMAC-192,**[<=]**

G2_N017.030.a.3 - (N017.030)a.3 K_Anwendungsspezifikation {K_Karte}, Option_kontaktlose_Schnittstelle
algorithmIdentifier = id-PACE-ECDH-GM-AES-CBC-CMAC-256,**[<=]**

G2_N017.030.b.1 - (N017.030)b.1 K_Anwendungsspezifikation {K_Karte}, Option_PACE_PCD
algorithmIdentifier = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128,**[<=]**

G2_N017.030.b.2 - (N017.030)b.2 K_Anwendungsspezifikation {K_Karte}, Option_PACE_PCD
algorithmIdentifier = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192,**[<=]**

G2_N017.030.b.3 - (N017.030)b.3 K_Anwendungsspezifikation {K_Karte}, Option_PACE_PCD
algorithmIdentifier = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256.**[<=]**

A_13968 - (N017.030)c K_TST

~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algorithmIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N017.031 - (N017.031) K_Anwendungsspezifikation {K_Karte}, Option_PACE_PCD
Wenn *algorithmIdentifier* Element der Menge {
id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128,
id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192,
id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256
} ist, genau dann MUSS ein symmetrisches Kartenverbindungsobjekt ein Attribut *accessRulesSessionkeys* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Kartenverbindungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in (N102.448) beschriebenen Use Cases.

G2_N017.032.a.1 - (N017.032)a.1 K_COS, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Zusätzlich zu den in (N017.032)a.2 gelisteten Kommandos MUSS ein symmetrisches Kartenverbindungsobjekt auch die folgenden Kommandos unterstützen:
i. Option_kontaktlose_Schnittstelle
GENERAL AUTHENTICATE (siehe (N085.001), (N085.003), (N085.005), (N085.007)),
ii. Option_PACE_PCD
GENERAL AUTHENTICATE (siehe (N085.031), (N085.033), (N085.035), (N085.037), (N085.039)).**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Kartenverbindungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.

G2_N017.032.a.2 - (N017.032)a.2 K_COS, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Zusätzlich zu den in (N017.032)a.1 gelisteten Kommandos MUSS ein symmetrischen Kartenverbindungsobjektauch die folgenden Kommandos unterstützen:
i. ACTIVATE (siehe (N034.814)),
ii. DEACTIVATE (siehe (N036.014)),
iii. DELETE (siehe (N037.144)),
iv. TERMINATE (siehe (N048.914)).**[<=]**

A_13969 - (N017.032)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für symmetrische Kartenverbindungsobjekte weitere Kommandos
1. unterstützt oder
2. ablehnt.**[<=]**

Hinweis CosH_34c: Absichtlich ist einem symmetrischen Kartenverbindungsobjekt kein Attribut shareable *zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein symmetrisches Kartenverbindungsobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.*

8.6.3 Privates Schlüsselobjekt

Hinweis CosH_1e5: Gegenüber der Generation 1 Spezifikation wird die dort beschriebene Unterteilung in private Authentisierungs-, Entschlüsselungs- und Signierobjekte in der Generation 2 aufgehoben. Mithin ist es möglich, dass ein privates Schlüsselobjekt sowohl für Authentisierungszwecke, als auch zur Entschlüsselung und zur Signaturerstellung nutzbar ist. Aus Sicherheitssicht ist es ratsam auf eine mögliche Vermischung wo irgend möglich zu verzichten. Die Trennung wird aufgehoben, da im Bereich IPSec oder auch TLS kryptographische Verfahren denkbar sind, wo ein privates Schlüsselobjekt sowohl entschlüsselt als auch signiert.

Ein privates Schlüsselobjekt wird

- zur Authentisierung einer Karte gegenüber einer anderen technischen Komponente
- zur Entschlüsselung von Daten und
- zur Berechnung elektronischer Signaturen

verwendet. Für private Schlüsselobjekte gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N017.100.a - (N017.100)a K_Anwendungsspezifikation {K_Karte}
Ein privates Schlüsselobjekt MUSS ein Attribut *keyIdentifier* mit einem ganzzahligen aus dem Intervall [2, 28] Wert besitzen.**[<=]**

A_13971 - (N017.100)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *keyIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N017.150 - (N017.150) K_Anwendungsspezifikation {K_Karte}
Ein privates Schlüsselobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100)a) besitzen.**[<=]**

G2_N017.200 - (N017.200) K_Anwendungsspezifikation {K_Karte}
Ein privates Schlüsselobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

G2_N017.300 - (N017.300) K_Anwendungsspezifikation {K_Karte}
Ein privates Schlüsselobjekt MUSS genau ein Attribut *privateKey* gemäß 8.2.3 besitzen.**[<=]**

(N017.310) Diese Anforderung ist absichtlich leer.

A_13972 - (N017.400) K_Anwendungsspezifikation {K_Karte}
Ein privates Schlüsselobjekt MUSS genau ein Attribut *listAlgorithmIdentifier* besitzen, welches angibt, für welche Zwecke das private Schlüsselobjekt verwendbar ist.**[<=]**

G2_N017.400.a - (N017.400)a K_Anwendungsspezifikation {K_Karte}
Die Liste *listAlgorithmIdentifier* MUSS eine Länge aus dem Intervall [1, 4] haben.**[<=]**

G2_N017.400.b - (N017.400)b K_Anwendungsspezifikation {K_Karte}
Jedes Listenelement in *listAlgorithmIdentifier* MUSS
1. genau einen *seIdentifier* gemäß (N007.900) und
2. genau eine Menge *setAlgorithmIdentifier* mit Elementen des Typs *algorithmIdentifier* enthalten.

3. Die Werte von *algorithmIdentifier* MÜSSEN aus den in (N017.600), (N017.900) und (N018.300) genannten Mengen gewählt werden.**[<=]**

G2_N017.400.c - (N017.400)c K_Anwendungsspezifikation {K_Karte}
Die Menge *setAlgorithmIdentifier* DARF NICHT mehr als sechs Elemente enthalten.**[<=]**

A_13973 - (N017.400)d K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS Listen des Typs
1. *listAlgorithmIdentifier*, welche mehr Elemente enthalten als nach (N017.400)a gefordert,
   i. unterstützt, oder
   ii. ablehnt.
2. *setAlgorithmIdentifier*, welche andere Elemente enthalten als nach (N017.400)b.3 gefordert,
   i. unterstützt, oder
   ii. ablehnt.
3. *setAlgorithmIdentifier*, welche mehr Elemente enthalten als nach (N017.400)c gefordert,
   i. unterstützt, oder
   ii. ablehnt.**[<=]**

G2_N017.420 - (N017.420) K_Anwendungsspezifikation {K_Karte}, Option_Kryptobox
Ist in *listAlgorithmIdentifier* ein *algorithmIdentifier* der Menge {~~rsaSessionkey4TC,~~ elcSessionkey4TC} enthalten, genau dann MUSS ein privates Authentisierungsobjekt ein Attribut *accessRulesSessionkeys* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

G2_N017.430 - (N017.430) K_Anwendungsspezifikation {K_Karte}
Ist in *listAlgorithmIdentifier* ein *algorithmIdentifier* der Menge {elcAsynchronAdmin} enthalten, genau dann MUSS ein privates Authentisierungsobjekt ein Attribut *numberScenario* vom Typ ganze Zahl aus dem Intervall [0, 32.767] besitzen.**[<=]**

(N017.500) Dieser Punkt ist absichtlich leer.

(N017.510) Dieser Punkt ist absichtlich leer.

G2_N017.600.a.1 - (N017.600)a.1 K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ *privateRsaKey* ist, dann MUSS zum Zwecke der Authentisierung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe CosT_7a2): {
   1. rsaClientAuthentication (siehe (N086.400) und (N086.900)b)
~~2. rsaRoleAuthentication, (siehe (N086.400) und (N086.900)c, Option_RSA_CVC)~~
~~3. rsaSessionkey4SM, (siehe (N086.400) und (N086.900)d, Option_DES)~~
~~4. rsaSessionkey4TC, (siehe (N084.402)c,(N086.902)c, Option_Kryptobox, Option_DES)~~
}.**[<=]**

G2_N017.600.a.2 - (N017.600)a.2 K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ *privateElcKey* ist, dann MUSS zum Zwecke der Authentisierung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe CosT_7a2): {
   1. elcAsynchronAdmin (siehe (N085.041) und (N085.068)),
   2. elcRoleAuthentication (siehe (N086.400) und (N086.900)a),
   3. elcSessionkey4SM (siehe (N085.054)),

4. elcSessionkey4TC (siehe (N085.056), Option_Kryptobox)
}.**[<=]**

A_13974 - (N017.600)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algorithmIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

(N017.610) ist absichtlich leer.

G2_N017.610.a - (N017.610)a K_Anwendungsspezifikation {K_Karte}, Option_DES, Option_RSA_CVC
~~Zusammenhang zwischen Moduluslänge und Card-to-Card-Algorithmen: Die *algorithmIdentifier* aus der Menge {rsaRoleAuthentication, rsaSessionkey4SM, rsaSessionkey4TC} DÜRFEN einem privaten Schlüssel NICHT zugewiesen werden, wenn dessen Moduluslänge ungleich 2048 bit ist.~~**[<=]**

A_13975 - (N017.610)b K_TST, Option_DES, Option_RSA_CVC
~~Die funktionale Eignung DARF einem Prüfling NICHT mit der Begründung verwehrt werden Werte *algorithmIdentifier* aus der Menge {rsaRoleAuthentication, rsaSessionkey4SM und rsaSessionkey4TC} würden für Moduluslängen ungleich 2048 bit
1. unterstützt oder
2. abgelehnt.~~**[<=]**

A_13976 - (N017.620)a K_Anwendungsspezifikation {K_Karte}
Zusammenhang zwischen Kurvenparametern und Card-to-Card-Algorithmen: Die *algorithmIdentifier* aus der Menge {elcRoleAuthentication, elcSessionkey4SM, elcSessionkey4TC} DÜRFEN einem privaten Schlüssel NICHT zugewiesen werden, wenn dessen *domainParameter* nicht in der Menge {brainpoolP256r1, brainpoolP384r1, brainpoolP512r1} enthalten ist.**[<=]**

A_13977 - (N017.620)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS Algorithmen aus der Menge {elcRoleAuthentication, elcSessionkey4SM, elcSessionkey4TC} für weitere *domainParameter*
1. unterstützt oder
2. ablehnt.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels der in (N100.900) oder (N102.900) beschriebenen Use Cases.

G2_N017.700 - (N017.700) K_COS
Ist einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus den in (N017.600)a.1 oder (N017.600)a.2 genannten Mengen zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:
~~1. EXTERNAL AUTHENTICATE (siehe (N084.400)c),~~
1. GENERAL AUTHENTICATE (siehe 15.4.4),
2. INTERNAL AUTHENTICATE (siehe (N086.400)),
3. PSO Compute Digital Signature (siehe (N088.600)c).**[<=]**

(N017.800) Dieser Punkt ist absichtlich leer.

G2_N017.900.a.1 - (N017.900)a.1 K_Anwendungsspezifikation {K_Karte}

Wenn das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ *privateRsaKey* ist, dann MUSS zum Zwecke der Entschlüsselung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe CosT_605): {
i. dieser Punkt ist absichtlich leer.
ii. rsaDecipherOaep (siehe (N089.200), (N092.800) und (N092.830))
}.**[<=]**

G2_N017.900.a.2 - (N017.900)a.2 K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ *privateElcKey* ist, dann MUSS zum Zwecke der Entschlüsselung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe CosT_605): {
i. elcSharedSecretCalculation (siehe (N089.800), (N093.600) und (N093.634))
}.**[<=]**

A_13978 - (N017.900)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algorithmIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in (N103.800) beschriebenen Use Cases.

G2_N018.000 - (N018.000) K_COS
Wenn einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus der in (N017.900)a.1 oder (N017.900)a.2 genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:
1. PSO Decipher (siehe (N089.200) und (N089.800)),
2. PSO Transcipher (siehe (N092.800), (N092.830), (N093.600) und (N093.634)).**[<=]**

(N018.100) Dieser Punkt ist absichtlich leer.

G2_N018.200 - (N018.200) K_Anwendungsspezifikation {K_Karte}
Ein privates Schlüsselobjekt MUSS ein Attribut *keyAvailable* vom Typ Boolean unterstützen:
a. Der Wert True zeigt an, dass das Attribut *privateKey* verwendbar ist.
b. Der Wert False zeigt an, dass das Attribut *privateKey* nicht verwendbar ist.
c. Auf dieses Attribut wird im Rahmen der folgenden Kommandos zugegriffen:
1. INTERNAL AUTHENTICATE (siehe (N086.810)),
2. PSO Compute Digital Signature (siehe (N088.500)),
3. PSO Decipher (siehe (N090.210)),
4. PSO Transcipher (siehe (N094.010)).**[<=]**

G2_N018.300.a.1 - (N018.300)a.1 K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ *privateRsaKey* ist, dann MUSS zum Zwecke der Signaturerzeugung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe CosT_c40): {
i. sign9796_2_DS2 (siehe (N088.000) und (N088.600)b),
ii. signPKCS1_V1_5 (siehe (N087.500) und (N088.600)d),
iii. signPSS (siehe (N087.500) und (N088.600)a)
}.**[<=]**

G2_N018.300.a.2 - (N018.300)a.2 K_Anwendungsspezifikation {K_Karte}

Wenn das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ *privateElcKey* ist, dann MUSS zum Zwecke der Signaturerzeugung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe CosT_c40): {
i. signECDSA (siehe (N087.500) und (N088.600)c)
}.**[<=]**

A_13979 - (N018.300)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algorithmIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in (N102.900) beschriebenen Use Cases.

G2_N018.400 - (N018.400) K_COS
Wenn einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus den in (N018.300)a.1 oder (N018.300)a.2 genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:
1. PSO Compute Digital Signature (siehe (N087.500) und (N088.000)).**[<=]**

Bevor eines der folgenden Kommando in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es selektieren. Dies geschieht entweder mittels des in (N102.900) beschriebenen Use Cases oder mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.

G2_N018.410 - (N018.410) K_COS
Ein privates Schlüsselobjekt MUSS folgendes Kommandos unterstützen:
1. GENERATE ASYMMETRIC KEY PAIR (siehe 14.9.3).**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.

G2_N018.420 - (N018.420) K_COS
Neben den in (N017.700), (N018.000), (N018.400) und (N18.410) genannten Kommandos MUSS ein privates Schlüsselobjekt zusätzlich die folgenden administrativen Kommandos unterstützen:
1. ACTIVATE (siehe (N034.814)),
2. DEACTIVATE (siehe (N036.014)),
3. DELETE (siehe (N037.114)),
4. TERMINATE (siehe (N048.914)).**[<=]**

A_13980 - (N018.422) K_TST
~~Ein privates Schlüsselobjekt KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für private Schlüsselobjekte weitere Kommandos
1. unterstützt oder
2. ablehnt.**[<=]**

Hinweis CosH_42f: Absichtlich ist einem privaten Schlüsselobjekt kein Attribut shareable *zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein privates Schlüsselobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.*

8.6.4 Öffentliches Schlüsselobjekt

Ein öffentliches Schlüsselobjekt wird als Oberbegriff für

- öffentliche Signaturprüfobjekte (siehe 8.6.4.1)
- öffentliche Authentisierungsobjekte (siehe 8.6.4.2)
- öffentliche Verschlüsselungsobjekte (siehe 8.6.4.3)

verwendet. Für öffentliche Schlüsselobjekte gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N018.500 - (N018.500) K_Anwendungsspezifikation {K_Karte}
Ein öffentliches Schlüsselobjekt MUSS ein Attribut *keyIdentifier* besitzen.**[<=]**

G2_N018.550 - (N018.550) K_Anwendungsspezifikation {K_Karte}
Eine öffentliches Schlüsselobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100)a) besitzen.**[<=]**

G2_N018.600 - (N018.600) K_Anwendungsspezifikation {K_Karte}
Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut *publicKey* gemäß 8.2.4 besitzen.**[<=]**

G2_N018.700 - (N018.700) K_Anwendungsspezifikation {K_Karte}
Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut vom Typ *oid* besitzen, welches angibt, für welche Zwecke es verwendbar ist.**[<=]**

G2_N018.800 - (N018.800) K_COS
Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem öffentlichen Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird.

G2_N018.820 - (N018.820) K_COS
Neben den in (N019.300), (N019.800) und (N019.826) genannten Kommandos MUSS ein öffentliches Schlüsselobjekt zusätzlich die folgenden administrativen Kommandos unterstützen:
1. ACTIVATE (siehe (N034.824)),
2. DEACTIVATE (siehe (N036.024)),
3. DELETE (siehe (N037.124)),
4. TERMINATE (siehe (N048.924)).**[<=]**

A_13982 - (N018.822) K_TST
~~Ein öffentliches Schlüsselobjekt KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für öffentliche Schlüsselobjekte weitere Kommandos
1. unterstützt oder
2. ablehnt.**[<=]**

(N018.900) Diese Anforderung ist absichtlich leer.

Hinweis CosH_c5d: Absichtlich ist einem öffentlichen Schlüsselobjekt kein Attribut shareable *zugeordnet (vergleiche etwa (N011.050)). Es wird erwartet, dass sich ein öffentliches Schlüsselobjekt in einem logischen Kanal unabhängig von beliebigen Aktivitäten in anderen Kanälen nutzen lässt.*

8.6.4.1 Öffentliches Signaturprüfobjekt

Ein öffentliches Signaturprüfobjekt wird zur Prüfung von Signaturen in einem CV-Zertifikat eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N019.000 - (N019.000) K_Anwendungsspezifikation {K_Karte}
Ein öffentliches Signaturprüfobjekt ist eine Erweiterung von einem öffentlichen Schlüsselobjekt und MUSS deshalb den Anforderungen aus 8.6.4 genügen.**[<=]**

G2_N019.100.a - (N019.100)a K_Anwendungsspezifikation {K_Karte}
Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge acht Oktett verwendet werden.**[<=]**

A_13983 - (N019.100)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Längen für *keyIdentifier* von öffentlichen Signaturprüfobjekten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N019.110.a - (N019.110)a K_Anwendungsspezifikation {K_Karte}
Ein öffentliches Signaturprüfobjekt MUSS
ein Attribute *accessRulesPublicSignatureVerificationObject* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen, das an Schlüsselobjekte weitergereicht wird, die mittels CV-Zertifikat importiert werden (siehe (N095.900)b.16..vii.A).**[<=]**

G2_N019.110.b - (N019.110)b K_Anwendungsspezifikation {K_Karte}
Ein öffentliches Signaturprüfobjekt MUSS
ein Attribute *accessRulesPublicAuthenticationObject* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen, das an Schlüsselobjekte weitergereicht wird, die mittels CV-Zertifikat importiert werden (siehe (N095.900)b.16.{vii.B, viii}).**[<=]**

(N019.200)a.1 ist absichtlich leer.

G2_N019.200.a.1 - (N019.200)a.1 K_Anwendungsspezifikation {K_Karte}, Option_RSA_CVC
~~Ist das Attribut *publicKey* des öffentlichen Signaturprüfobjektes vom Typ *publicRsaKey* MUSS der Wert von *oid* Element der Menge {~~
~~1. sigS_ISO9796-2Withrsa_sha256~~
~~} sein, siehe CosT_a91.~~**[<=]**

G2_N019.200.a.2 - (N019.200)a.2 K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *publicKey* des öffentlichen Signaturprüfobjektes vom Typ *publicElcKey* ist, dann MUSS der Wert von *oid* Element der Menge {
1. ecdsa-with-SHA256,
2. ecdsa-with-SHA384,
3. ecdsa-with-SHA512

} sein (siehe CosT_a91) und zu den Domainparametern von *publicElcKey* passen.**[<=]**

A_13984 - (N019.200)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *oid* eines öffentlichen Signaturprüfobjektes
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N019.210.a - (N019.210)a K_Anwendungsspezifikation {K_Karte}

Wenn das Attribut *publicKey* vom Typ *publicElcKey* ist, dann gilt: Das öffentliche Signaturprüfobjekt MUSS ein Attribut CHAT enthalten, welches Aktionen kennzeichnet, die nach einer erfolgreichen Authentisierung mit einem öffentlichen Authentisierungsobjekt freigeschaltet werden.**[<=]**

G2_N019.210.b - (N019.210)b K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *publicKey* vom Typ *publicElcKey* ist, dann gilt: Das öffentliche Signaturprüfobjekt MUSS ein Attribut *expirationDate* vom Typ *date* (siehe (N008.120)) enthalten, welches angibt, ab welchem Zeitpunkt die Schlüsselverwendung endet.**[<=]**

Bevor eines der folgenden Kommando in der Lage ist mit dem öffentlichen Signaturprüfobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in (N103.300) beschriebenen Use Cases.

G2_N019.300 - (N019.300) K_COS
Ein öffentliches Signaturprüfobjekt MUSS zusätzlich zu den in (N018.820) genannten Kommandos die folgenden Kommandos unterstützen:
1. PSO Verify Certificate (siehe ~~(N095.100) und~~ (N095.500),
2. PSO Verify Digital Signature (siehe (N096.388)).**[<=]**

8.6.4.2 Öffentliches Authentisierungsobjekt

Ein öffentliches Authentisierungsobjekt wird zur Authentisierung einer anderen technischen Komponente gegenüber dieser Komponente eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N019.400 - (N019.400) K_Anwendungsspezifikation {K_Karte}
Ein öffentliches Authentisierungsobjekt ist eine Erweiterung von einem öffentlichen Schlüsselobjekt und MUSS deshalb den Anforderungen aus 8.6.4 genügen.**[<=]**

G2_N019.500.a - (N019.500)a K_Anwendungsspezifikation {K_Karte}
Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge zwölf verwendet werden.**[<=]**

A_13985 - (N019.500)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Längen für *keyIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

(N019.600)a.1 ist absichtlich leer.

G2_N019.600.a.1 - (N019.600)a.1 K_Anwendungsspezifikation {K_Karte}
~~Ist das Attribut *publicKey* des öffentlichen Authentisierungsobjekte vom Typ *publicRsaKey* MUSS der Wert von *oid* Element der Menge {~~
~~1. authS_ISO9796-2Withrsa_sha256_mutual~~
~~} sein, siehe CosT_a91.~~**[<=]**

G2_N019.600.a.2 - (N019.600)a.2 K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *publicKey* des öffentlichen Authentisierungsobjekte vom Typ *publicElcKey* ist, dann MUSS der Wert von *oid* Element der Menge {
1. authS_gemSpec-COS-G2_ecc-with-sha256,
2. authS_gemSpec-COS-G2_ecc-with-sha384,
3. authS_gemSpec-COS-G2_ecc-with-sha512

} sein (siehe CosT_a91) und zu den Domainparametern von *publicElcKey* passen.**[<=]**

A_13991 - (N019.600)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *oid* eines öffentlichen Authentisierungsobjekte
1. unterstützt oder
2. ablehnt.**[<=]**

(N019.700)a ist absichtlich leer.

G2_N019.700.a - (N019.700)a K_Anwendungsspezifikation {K_Karte}, Option_RSA_CVC
~~Ist das Attribut *publicKey* des öffentlichen Authentisierungsobjekte vom Typ *publicRsaKey* MUSS das öffentliche Authentisierungsobjekt ein Attribut *accessRights* besitzen, welches eine *CHA* gemäß 7.1.1.4 speichert.~~**[<=]**

G2_N019.700.b - (N019.700)b K_Anwendungsspezifikation {K_Karte}
Wenn das Attribut *publicKey* des öffentlichen Authentisierungsobjekte vom Typ *publicElcKey* ist, dann MUSS das öffentliche Authentisierungsobjekt ein Attribut
1. *accessRights* besitzen, welches ein CHAT speichert und
2. *expirationDate* vom Typ *date* (siehe (N008.120)) enthalten, welches angibt, ab welchem Zeitpunkt die Schlüsselverwendung endet.**[<=]**

Bevor eines der folgenden Kommandos in der Lage ist mit dem öffentlichen Authentisierungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in (N101.900) beschriebenen Use Cases.

G2_N019.800 - (N019.800) K_COS
Ein öffentliches Authentisierungsobjekt MUSS zusätzlich zu den in (N018.820) genannten Kommandos die folgenden Kommandos unterstützen:
a. EXTERNAL AUTHENTICATE (siehe (N083.500) und (N084.400)),
b. GENERAL AUTHENTICATE (siehe 15.4.4).
~~c. INTERNAL AUTHENTICATE (siehe (N086.400), (N086.900)d und (N086.902)c).~~**[<=]**

8.6.4.3 Öffentliches Verschlüsselungsobjekt, Option_Kryptobox

Ein öffentliches Verschlüsselungsobjekt wird zum Verschlüsseln von Daten eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

G2_N019.820 - (N019.820) K_Anwendungsspezifikation {K_Karte}, Option_Kryptobox
Ein öffentliches Verschlüsselungsobjekt ist eine Erweiterung von einem öffentlichen Schlüsselobjekt und MUSS deshalb den Anforderungen aus 8.6.4 genügen.**[<=]**

G2_N019.822.a - (N019.822)a K_Anwendungsspezifikation {K_Karte }, Option_Kryptobox
Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge zwölf verwendet werden.**[<=]**

A_13992 - (N019.822)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Längen für *keyIdentifier*
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N019.824.a.1 - (N019.824)a.1 K_Anwendungsspezifikation {K_Karte}, Option_Kryptobox

Wenn das Attribut *publicKey* des öffentlichen Verschlüsselungsobjektes vom Typ *publicRsaKey* ist, dann MUSS der Wert von *oid* Element der Menge {
i. id-RSAES-OAEP
} sein, siehe CosT_a91.**[<=]**

G2_N019.824.a.2 - (N019.824)a.2 K_Anwendungsspezifikation {K_Karte}, Option_Kryptobox
Wenn das Attribut *publicKey* des öffentlichen Verschlüsselungsobjektes vom Typ *publicElcKey* ist, dann MUSS der Wert von *oid* Element der Menge {
i. id-ELC-shared-secret-calculation
} sein, siehe CosT_a91.**[<=]**

A_13993 - (N019.824)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *oid* eines öffentlichen Verschlüsselungsobjektes
1. unterstützt oder
2. ablehnt.**[<=]**

Bevor eines der folgenden Kommando in der Lage ist mit dem öffentlichen Signaturprüfobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in (N103.850) beschriebenen Use Cases.

G2_N019.826 - (N019.826) K_COS, Option_Kryptobox
Ein öffentliches Verschlüsselungsobjekt MUSS zusätzlich zu den in (N018.820) genannten Kommandos die folgenden Kommandos unterstützen:
1. PSO Encipher (siehe (N091.424), (N091.434) und (N103.850)).**[<=]**

8.7 Datenobjekte (informativ)

Auf Datenobjekte wird im Rahmen der [ISO/IEC 7816-4] Kommandos GET DATA und PUT DATA zugegriffen. Da diese Kommandos für diese Version der Spezifikation nicht verpflichtend sind, werden sie hier nicht weiter behandelt.

8.8 Security Environment (informativ)

Zum Begriff "Security Environment" gemäß [ISO/IEC 7816-4#10.3.3] existiert eine große Bandbreite an Interpretationen. Deshalb wird versucht, im normativen Bereich dieses Dokumentes auf diesen problematischen Begriff möglichst zu verzichten. Aus demselben Grund wird in diesem informativen (das heißt nicht-normativen) Kapitel nur das weitgehend unstrittige Prinzip der Security Environments kurz skizziert.

Security Environments eröffnen die Möglichkeit, gewisse Attribute von Objekten in Abhängigkeit von der Einsatzumgebung mit konkreten Werten zu versehen. In diesem Dokument wird davon beim Attributstyp *interfaceDependentAccessRules* (siehe 8.1.4) Gebrauch gemacht. Damit ergeben sich beim Design von Anwendungen Gestaltungsmöglichkeiten.

Beispiel: Eine Karte wird in zwei verschiedenen Umgebungen eingesetzt und in beiden sind sowohl vertrauliche Daten (EF.Notfalldaten) als auch allgemein bekannte Daten (EF.X509_Zertifikat) auslesbar (READ BINARY).

1. In der Umgebung_1 werden Lauschangriffe auf die physikalische Schnittstelle (siehe CosA_e09) befürchtet. Zur Abwehr derartiger Angriffe wird die Zugriffsregel so gewählt, dass sensitive Daten nur vertraulich und integer übertragbar sind (siehe (N022.600)c). Für allgemein bekannte Daten wird ein derartiger Schutz nicht verlangt.
2. In der Umgebung_2 existieren keine Angriffe auf die physikalische Schnittstelle (siehe CosA_e09). Deshalb wird aus diversen Gründen die Zugriffsregel so gewählt, dass alle Daten ungeschützt übertragen werden.

Zu jeder Einsatzumgebung existiert ein Satz Zugriffsregeln:

3. Umgebung_1:
   a. EF.Notfalldaten READ BINARY an einem Kiosk mit Secure Messaging
   b. EF.X509_Zertifikat READ BINARY immer erlaubt
4. Umgebung_2:
   a. EF.Notfalldaten READ BINARY in einer Arztpraxis ohne Verschlüsselung
   b. EF.X509_Zertifikat READ BINARY immer erlaubt

Welcher Satz von Zugriffsregeln gilt, wird durch die Einsatzumgebung bestimmt. Eine konkrete Einsatzumgebung (ein konkretes Security Environment) wird durch den Use Case in (N099.900) ausgewählt.

Auf der anderen Seite werden Security Environments verwendet, um Schlüssel und andere Parameter für kryptographische Operationen auszuwählen. Die Vorgehensweise ist hier anders als bei Passwörtern. Welches Passwort von einem Kommando aus 14.6 betroffen ist, bestimmt ein Parameter dieses Kommandos.

Die Vorgehensweise ist für Schlüssel eher vergleichbar mit Dateien. Welche Datei von einem Kommando betroffen ist, bestimmt das Attribut *currentEF*, das mittels SELECT-Kommando vor Ausführung dieses Kommandos passend einzustellen ist. Weil die dateiorientierten Kommandos lediglich in der Lage sind stets nur mit einer Datei pro Kommando zu arbeiten, ist ein einziges Attribut *currentEF* pro Ordner völlig ausreichend (siehe (N029.900)m).

Bei einigen kryptographischen Kommandos ist es hingegen möglich, dass bei der Bearbeitung mehrere kryptographische Objekte beteiligt sind, wie etwa

- Sessionkeys für Vertraulichkeit,
- Sessionkeys für Authentizität,
- Use Case spezifischer Schlüssel, etwa PSO Compute Digital Signature.

Statt nun jedem derartigen Kommando analog zu den Passwortkommandos alle Schlüsselparameter mitzuliefern, ist in [ISO/IEC 7816-4] entschieden worden, diese Parameter zuvor mittels MANAGE SECURITY ENVIRONMENT Kommando (siehe 14.9.9) auszuwählen. Derartige Selektionen wirken sich im Datenmodell dieses Dokumentes auf das Attribut *keyReferenceList* (siehe (N029.900)c) aus.

Hinweis CosH_981: Die normativen Regeln dieses Dokumentes sind so gewählt, dass es für eine performante Implementierung zulässig ist, die kanalspezifischen Attribute (siehe 12.1) im RAM zu speichern. Die genannten Attribute wurden lediglich der einfacheren Darstellung wegen, wie gezeigt zugeordnet.

Hinweis CosH_2ab: Die normativen Regeln dieses Dokumentes sind so gewählt, dass es für eine speicherplatzoptimale Implementierung zulässig ist, die kanalspezifischen Ordnerattribute in (N030.000) nur für currentFolder *und allen seinen Vorfahren inklusive* root *zu speichern, anstatt für alle Ordner innerhalb des Objektsystems. Die genannten Attribute wurden lediglich der einfacheren Darstellung wegen, jedem Ordner zugeordnet.*

8.9 Sicherheitsstatus (informativ)

Zum Begriff "Sicherheitsstatus" gemäß [ISO/IEC 7816-4#9] existiert eine große Bandbereite an Interpretationen. Deshalb wird versucht auf diesen problematischen Begriff im normativen Bereich dieses Dokumentes möglichst zu verzichten. Aus demselben Grund wird in diesem informativen (das heißt nicht-normativen) Kapitel nur das weitgehend unstrittige Prinzip des Sicherheitsstatus kurz skizziert.

Gemäß [ISO/IEC 7816-4] zählen zum Sicherheitsstatus

1. Globale Sicherheitsstatus, welche durch Benutzerverifikation oder Komponentenauthentikation mit Objekten im Ordner root (siehe (N019.900)a) modifizierbar ist.
2. Applikationsspezifischer Sicherheitsstatus, welche mit Benutzerverifikationen oder Komponentenauthentisierungen mit Objekten in beliebigen Ordnern modifizierbar ist.
3. File-spezifischer Status, welcher in diesem Dokument unberücksichtigt bleibt.
4. Kommando-spezifischer Status, welchem bei der Auswertung von Zugriffsregeln Rechnung getragen wird.

In diesem Dokument wird bezüglich der globalen und applikationsspezifischen Status für alle Ordner ein Sicherheitsstatus angenommen und für die

5. Benutzerverifikation durch (N029.900)i bis (N029.900)k Rechnung getragen, wobei dort auch die kleinste Anzahl der zu unterstützenden Sicherheitsstatus für Benutzerverifikation festgelegt wird.
6. Komponentenauthentisierung durch (N029.900)e bis (N029.900)h Rechnung getragen, wobei dort auch die kleinste Anzahl der zu unterstützenden Sicherheitsstatus für Komponentenauthentisierung festgelegt wird.

Hinweis CosH_517: Die normativen Regeln dieses Dokumentes sind so gewählt, dass es für eine performante Implementierung zulässig ist, die Attribute globalSecurityList, dfSpecificSecurityList, globalPasswordList *und* dfSpecificPasswordList *im RAM zu speichern. Die genannten Attribute wurden lediglich der einfacheren Darstellung wegen wie dargestellt zugeordnet.*

9 Objektsystem (normativ)

Eine Smartcard wird in diesem Dokument als *sicherer Datenspeicher* betrachtet, wobei die Betonung auf beiden Wörtern liegt.

- Datenspeicher: Eine Smartcard speichert beliebige Informationen ganz analog zur Festplatte eines Computers in Dateien (siehe 8.3.2).
- Sicherheit: Der Zugriff auf die in Dateien gespeicherten Informationen wird durch Regeln festgelegt. Die Einhaltung der Zugriffsregeln wird durch das Betriebssystem gewährleistet. Typischerweise enthalten Regeln Zugriffsbeschränkungen dergestalt, dass erst nach erfolgreicher Benutzerverifikation oder Komponentenauthentisierung ein Zugriff gestattet ist. Darüber hinaus wird vielfach zusätzlich ein vertraulicher und authentischer Datenaustausch erzwungen.

Typischerweise werden Dateien und damit Informationen hierarchisch strukturiert.

Die Regeln in diesem Dokument sind so aufgebaut, dass sich ein hierarchisches System mit mindestens vier Ordnerebenen aufbauen lässt (*root* enthält DF2, DF2 enthält DF3, DF3 enthält DF4, DF4 enthält keinen weiteren Ordner).

Private und symmetrische Schlüssel sowie Passwörter lassen sich per "backtracking" suchen. Das bedeutet, sie werden zunächst im aktuellen Verzeichnis gesucht. Wenn die Suche dort erfolglos ist, dann wird rekursiv in der nächsthöheren Ebene gesucht.

Eine DF-spezifische Suche bezieht *root* nie mit ein.

Öffentliche Schlüssel werden als zentral gespeicherte Objekte betrachtet. In der Regel gehören solche Schlüssel einer externen Komponente und werden mittels eines Zertifikates importiert (siehe 14.8.7).

9.1 Aufbau und Strukturtiefe

Dieses Kapitel legt die hierarchische Struktur fest, so wie sie an der Schnittstelle gesehen wird. Wie die Information und die Struktur kartenintern gespeichert werden, ist nicht Gegenstand dieses Dokumentes. Zudem wird hier nicht der sonst übliche Terminus "Filesystem" verwendet, sondern "Objektsystem", weil neben Files auch andere Objekttypen, wie Passwörter und Schlüssel, als eigenständige Artefakte betrachtet werden.

Bei der Spezifikation von Anwendungen sind folgende Regeln einzuhalten:

A_13996 - (N019.900) K_COS
Das COS MUSS ein hierarchisches Objektsystem mit mehreren Ebenen unterstützen.**[<=]**

G2_N019.900.a - (N019.900)a K_Anwendungsspezifikation {K_Karte}
Das Objektsystem MUSS ein Attribut *root* mit Eigenschaften gemäß (N019.910) und (N019.920) besitzen.**[<=]**

A_13997 - (N019.900)b K_Anwendungsspezifikation {K_Karte}

Das Objektsystem MUSS die Attribute *coldAnswerToReset* und *warmAnswerToReset* vom Typ Oktettstring besitzen für die in (N024.100) Anforderungen genannt werden.**[<=]**

G2_N019.900.b.1 - (N019.900)b.1 K_COS
Das COS MUSS das Attribut *coldAnswerToReset* im Rahmen eines Cold Reset gemäß (N023.920)b versenden.**[<=]**

G2_N019.900.b.2 - (N019.900)b.2 K_COS
Das COS MUSS das Attribut *warmAnswerToReset* im Rahmen eines Warm Reset gemäß (N023.920)c versenden.**[<=]**

A_13998 - (N019.900)b.3 K_Anwendungsspezifikation {K_Karte}
Es MÜSSEN identische Anforderungen an die Attribute *coldAnswerToReset* und *warmAnswerToReset* gestellt werden.**[<=]**

A_13999 - (N019.900)b.4 K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für die Attribute *coldAnswerToReset* und *warmAnswerToReset* unterschiedliche Werte verwendet.**[<=]**

G2_N019.900.c - (N019.900)c K_Anwendungsspezifikation {K_Karte}
Das Objektsystem MUSS ein Attribut *iccsn8* vom Typ Oktettstring der Länge acht Oktett und beliebigem Inhalt besitzen.**[<=]**

A_14000 - (N019.900)d K_COS
Das Objektsystem MUSS ein Attribut *applicationPublicKeyList* mit folgenden Eigenschaften unterstützen:
1. In *applicationPublicKeyList* sind beliebig viele Listenelemente möglich.
2. Als Listenelemente wird der Objekttyp "Öffentliches Schlüsselobjekt" (siehe8.6.4) unterstützt in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).
3. Alle Elemente der Liste *applicationPublicKeyList* werden persistent gespeichert.
4. Elemente der Liste *applicationPublicKeyList* werden
   i. entweder zentral an einer Stelle,
   ii. oder dezentral, etwa über mehrere Ordner verteilt
   gespeichert.**[<=]**

A_14001 - (N019.900)e K_COS
Das Objektsystem MUSS ein Attribut *persistentCache* unterstützen.**[<=]**

G2_N019.900.e.1 - (N019.900)e.1 K_COS
Das COS MUSS eine beliebige Anzahl Listenelemente in *persistentCache* unterstützen.**[<=]**

G2_N019.900.e.2 - (N019.900)e.2 K_Anwendungsspezifikation {K_Karte}
Die Anwendungsspezifikation ~~SOLL~~ MUSS eine Anzahl an Listenelementen für *persistentCache* vorschreiben, die mindestens zu unterstützen ist.**[<=]**

G2_N019.900.e.3 - (N019.900)e.3 K_COS
Das COS MUSS in *persistentCache* als Listenelemente die Objekttypen "Öffentliches Signaturprüfobjekt" (siehe8.6.4.1) und "Öffentliches Authentisierungsobjekt" (siehe8.6.4.2) unterstützen in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).**[<=]**

G2_N019.900.e.4 - (N019.900)e.4 K_COS
Das COS MUSS alle Listenelemente in *persistentCache* persistent speichern.**[<=]**

A_14002 - (N019.900)e.5 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS Elemente der List *persistentCache*
i. zentral an einer Stelle,
ii. oder dezentral über mehrere Ordner verteilt
speichert.**[<=]**

G2_N019.900.f - (N019.900)f K_Anwendungsspezifikation {K_Karte}
Die Anwendungsspezifikation ~~SOLL~~ MUSS eine Anzahl an Listenelementen für *persistentPublicKeyList* vorschreiben, die mindestens zu unterstützten ist.**[<=]**

A_14003 - (N019.900)g K_COS
Das Objektsystem MUSS ein Attribut *volatileCache* vom Typ Liste unterstützen.**[<=]**

G2_N019.900.g.1.i - (N019.900)g.1 K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS Listenlängen mit dem Wert null, eins oder größer unterstützt.**[<=]**

G2_N019.900.g.2 - (N019.900)g.2 K_COS
Das COS MUSS in *volatileCache* als Listenelemente die Objekttypen "Öffentliches Signaturprüfobjekt" (siehe8.6.4.1) und "Öffentliches Authentisierungsobjekt" (siehe8.6.4.2) unterstützen in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).**[<=]**

A_14004 - (N019.900)g.3 K_COS
Das COS DARF Elemente der Liste *volatileCache* NICHT persistent speichern.**[<=]**

A_14005 - (N019.900)g.4 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS Elemente der List `volatileCache`
i. zentral an einer Stelle,
ii. oder dezentral über mehrere Ordner verteilt
speichert.**[<=]**

G2_N019.900.h - (N019.900)h K_COS, Option_logische_Kanäle
Das COS MUSS alle Elemente der Liste *allPublicKeyList* in allen logischen Kanälen shareable zur Verfügung stellen.**[<=]**

G2_N019.900.i - (N019.900)i K_Anwendungsspezifikation {K_Karte}
Das Objektsystem MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe (N007.100)a) besitzen.**[<=]**

G2_N019.900.j - (N019.900)j K_Anwendungsspezifikation {K_Karte}
Das Objektsystem MUSS genau ein Attribut *pointInTime* vom Typ *date* gemäß (N008.120) besitzen.**[<=]**

Hinweis CosH_c83: Der zu unterstützende Wertebereich von pointInTime *ergibt sich implizit aus dem Attribut CED, welches in CV-Zertifikaten enthalten ist, siehe [gemSpec_PKI#6.7.2.6].*

A_14006 - (N019.910) K_Anwendungsspezifikation {K_Karte}
Für das Attribut *root* des Objektsystems MUSS gelten:
a. Die Anwendungsspezifikation lässt für *root* die Wahl zwischen den Typen
   1. Applikation (siehe 8.3.1.1) und
   2. Application Dedicated File (siehe 8.3.1.3) zu.
b. Die Anwendungsspezifikation fordert als Zugriffbedingung für die Zugriffsart DELETE (siehe (N037.100)) den Wert NEVER (siehe (N022.100)).**[<=]**

G2_N019.920.a - (N019.920)a K_COS
Das COS MUSS für *root* den Typ Applikation (siehe 8.3.1.1) unterstützen.**[<=]**

A_14007 - (N019.920)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für *root* weitere Ordnertypen
1. unterstützt oder
2. ablehnt.**[<=]**

A_14008 - (N019.920)c K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS für *root* weitere Zugriffsbedingungen für die Zugriffsart DELETE
1. unterstützt oder
2. ablehnt.**[<=]**

A_14009 - (N019.930) K_Anwendungsspezifikation {K_Karte}
Die Werte aller Attribute *applicationIdentifier* MÜSSEN paarweise verschieden sein.~~Im Objektsystem DARF es KEINE zwei verschiedenen Ordner geben, deren Attribut *applicationIdentifier* identisch ist.~~ **[<=]**

G2_N020.000 - (N020.000) K_COS
Der Ordner *root* wird der Ebene_0 zugeordnet. Ebene_0 wird als "höchste Ebene im Objektsystem" bezeichnet.**[<=]**

G2_N020.100 - (N020.100) K_COS
Wenn ein Ordner der Ebene_i zugeordnet ist, dann werden alle Elemente der Liste *children* dieses Ordners der Ebene_(i + 1) zugeordnet. Ebene_i ist die nächsthöhere Ebene zu Ebene_(i + 1). Ebene_(i + 1) ist die nächsttiefere Ebene zu Ebene_i.**[<=]**

(N020.200) Diese Anforderung ist absichtlich leer.

G2_N020.300 - (N020.300) K_COS
Ordner, die der Ebene_0 (*root*), Ebene_1 oder Ebene_2 zugeordnet sind, MÜSSEN im Attribut *children* als Listenelement die Objekttypen Applikation (siehe 8.3.1.1) und Dedicated File (siehe 8.3.1.2) und Application Dedicated File (siehe 8.3.1.3) zulassen.**[<=]**

A_14010 - (N020.390) K_Anwendungsspezifikation {K_Karte}
Ordner, die der Ebene_3 zugeordnet sind DÜRFEN KEIN Objekt des Typs Ordner enthalten.**[<=]**

A_14011 - (N020.400) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS in Ordnern, die der Ebene_3 zugeordnet sind, im Attribut *children* Ordner
a. zulässt oder
b. ablehnt.**[<=]**

Dem Objektsystem sind weitere, kanalspezifische Attribute zugeordnet, die einem flüchtig (im RAM) gespeicherten Kanalkontext (siehe (N029.900)) zugerechnet werden. Sie werden im Rahmen einer Selektion des Ordners verändert.

Gemäß 8.3 (File), 8.4 (Reguläres-Passwortobjekt), 8.5 (Multireferenz-Passwortobjekt) und 8.6 (Schlüsselobjekt) besitzen die dort genannten Objekttypen ein Attribut *lifeCycleStatus*. Gemäß (N020.000) und (N020.100) lassen sich die Instanzen aller Objekttypen Ebenen zuordnen, wobei Rekords hier als eine nächsttiefere Ebene zu der Datei aufgefasst werden, in der sie enthalten sind. Es wird nun unterschieden zwischen dem physikalischen und dem logischen Wert des *lifeCycleStatus*:

A_14012 - (N020.500) K_COS
Als physikalischer Wert des *lifeCycleStatus* MUSS der Wert betrachtet werden, den das entsprechende Attribut eines Objektes besitzt.**[<=]**

A_14013 - (N020.600) K_COS
Als logischer Wert des *lifeCycleStatus* MUSS der Wert betrachtet werden, der sich aus der Betrachtung des physikalischen Wertes des *lifeCycleStatus* eines Objektes sowie der logischen Werte des *lifeCycleStatus* in allen höheren Ebenen ergibt. Dabei gilt folgende Rangordnung und Rekursion:**[<=]**

G2_N020.600.a - (N020.600)a K_COS
"Operational state (active)" < "Operational state (deactivated)" < "Termination state"**[<=]**

G2_N020.600.b - (N020.600)b K_COS
Für das Objekt *root* MUSS als logischer Wert der nächsthöheren Ebene der physikalische Wert des *lifeCycleStatus* des Objektsystems verwendet werden (siehe (N019.900)i)**[<=]**

G2_N020.600.c - (N020.600)c K_COS
Für ein Objekt mit einem Attribut *lifeCycleStatus* MUSS gelten: Wenn

1. der logische Wert des *lifeCycleStatus* der nächsthöheren Ebene größer ist als der physikalische Wert des *lifeCycleStatus* des Objektes, dann ist der logische Wert des *lifeCycleStatus* des Objektes gleich dem logischen Wert des *lifeCycleStatus* der nächsthöheren Ebene.
2. sonst ist der der logische Wert des *lifeCycleStatus* des Objektes gleich seinem physikalischen.

[<=]

G2_N020.600.d - (N020.600)d K_COS
Für ein Objekt, welches selbst kein Attribut *lifeCycleStatus* besitzt, MUSS der logische Wert von *lifeCycleStatus* identisch zum logischen Wert der nächsthöheren Ebene sein.**[<=]**

9.2 Objektsuche

Gemäß 9.1 ist an der Schnittstelle zur Karte ein hierarchisches Objektsystem sichtbar. Dieses Kapitel legt fest, wie ein Objekt gesucht und nach welchen Regeln es gefunden wird.

9.2.1 Filesuche

Eine Suche nach Files (das heißt Ordnern und Dateien) findet lediglich im Rahmen einer Selektion statt. Deshalb ist die explizite Suche in 14.2.6 beschrieben. Daneben unterstützen die Kommandos, welche sich auf Dateien beziehen, Varianten mit Parameter *shortFileIdentifier*. Bei den jeweiligen Kommandos ist beschrieben, wie in diesem Fall eine Datei gesucht wird.

9.2.2 Suche nach einem Passwortobjekt

Passwörter werden im Rahmen der Kommandos in 14.6 verwendet, aber auch bei der Auswertung von Zugriffsregeln (siehe (N022.200)). Bei der Passwortsuche handelt es sich um eine karteninterne Funktionalität, bei der viele Implementierungsdetails eine

Rolle spielen. An der Schnittstelle "Interpreter" (siehe CosA_e09) wird das im Folgenden festgelegte Verhalten deutlich:

G2_N020.700 - (N020.700) K_COS

Tabelle 63: CosT_65c: Definition der Funktion SearchPwd(...)

Input:	*passwordReference*	Gemäß (N072.800)
	startFolder	Ordner, bei dem die Suche nach einem Passwortobjekt startet
Output:	*password*	Enthält das gefundene Passwortobjekt
Errors:	*pwdNotFound*	Zu den gegebenen Inputparametern wurde kein passendes Passwortobjekt gefunden.
Notation:		*password* = SearchPwd(*startFolder*, *passwordReference*)

Das COS MUSS den Inputparameter *passwordReference* gemäß (N072.800) wie folgt in seine Bestandteile zerlegen:
a. *identifier* = *passwordReference* mod 128
b. *location* = *passwordReference* – *identifier***[<=]**

A_14131 - (N020.800) K_COS
Wenn *location* den Wert '00' = 0 besitzt und damit ein globales Passwort adressiert, dann MUSS das COS im Wurzelverzeichnis *root* des Objektsystems nach einem Passwort suchen, dessen Attribut *pwdIdentifier* identisch zu *identifier* ist.**[<=]**

G2_N020.800.a - (N020.800)a K_COS
Wenn ein solches Passwortobjekt existiert,
dann MUSS das COS als Outputparameter *password* verwenden.**[<=]**

G2_N020.800.b - (N020.800)b K_COS
Wenn ein solches Passwortobjekt nicht existiert,
dann MUSS das COS den Algorithmus mit *pwdNotFound* abbrechen.**[<=]**

G2_N020.900 - (N020.900) K_COS
Wenn *location* den Wert '80' = 128 besitzt und damit ein DF-spezifisches Passwort adressiert,
dann MUSS das COS die "lokale Variable "*folder* auf *startFolder* setzen.**[<=]**

G2_N021.000.a - (N021.000)a K_COS
Wenn *folder* auf das Wurzelverzeichnis *root* zeigt,
dann MUSS das COS den Algorithmus mit *pwdNotFound* abbrechen.**[<=]**

A_14132 - (N021.000)b K_COS
Wenn *folder* nicht auf das Wurzelverzeichnis *root* zeigt,
dann MUSS das COS in *folder* nach einem Passwort suchen, dessen Attribut *pwdIdentifier* identisch zu *identifier* ist.**[<=]**

G2_N021.000.b.1 - (N021.000)b.1 K_COS
Wenn ein solches Passwortobjekt existiert,
dann MUSS das COS es als Outputparameter *password* verwenden.**[<=]**

G2_N021.000.b.2 - (N021.000)b.2 K_COS

Wenn ein solches Passwortobjekt nicht existiert,
dann MUSS das COS *folder* auf den nächsthöheren Ordner setzen und den Algorithmus mit Schritt (N021.000)a fortsetzen.**[<=]**

9.2.3 Suche nach einem Schlüsselobjekt

Die hier beschriebene Funktion ist eine Generalisierung der anderen Schlüsselsuchfunktionen und stellt damit den generalisierten Einsprungspunkt für die Schlüsselsuche dar, so wie sie an verschiedenen Stellen in diesem Dokument verwendet wird.

Tabelle 64: CosT_a08: Definition der Funktion SearchKey(...)

Input:	*startFolder*	Ordner, aus dem heraus die Suche startet
	identifier	Für diesen Parameter sind die folgende Wertebereiche möglich: a) ein Oktett *keyReference* gemäß (N099.600), b) acht Oktett *keyIdentifier* gemäß (N019.100)a, c) zwölf Oktett *keyIdentifier* gemäß (N019.500)a, d) zwölf Oktett *keyIdentifier* gemäß (N019.822)a, e) herstellerspezifisch, zur Auswahl von Sessionkeys
	algID	Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt, oder "Wildcard"
Output:	*key*	Enthält das gefundene Schlüsselobjekt
Errors:	*keyNotFound*	Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden.
	notSupported	Der Schlüssel unterstützt den von *algID* geforderten Algorithmus nicht.
Notation:		*key* = SearchKey(*startFolder*, *identifier*, *algID*)

A_14133 - (N021.050)a K_COS
Enthält der Parameter *identifier* eine Schlüsselreferenz gemäß (N099.600) (d. h. symmetrisches Authentisierungsobjekt, symmetrisches Kartenverbindungsobjekt oder privates Schlüsselobjekt), dann MUSS das COS folgende Funktion nutzen:
key = SearchSecretKey(*startFolder*, *identifier*, *algID*) **[<=]**

A_14134 - (N021.050)b K_COS
Enthält der Parameter *identifier* eine Schlüsselreferenz gemäß (N019.100)a oder (N019.500)a oder (N019.822)a (d. h. ein öffentliches Schlüsselobjekt), dann MUSS das COS folgende Funktion nutzen:
key = SearchPublicKey(*startFolder*, *identifier*, *algID*) **[<=]**

G2_N021.050.c - (N021.050)c K_COS
Enthält der Parameter *identifier* eine Schlüsselreferenz gemäß eines herstellerspezifischen Wertebereiches, dann gilt: Es handelt sich um Sessionkeys, die im Rahmen von PSO-Kommandos zur Unterstützung eines Trusted Channels eingesetzt werden. Solange der Sicherheitszustand, der im Rahmen der Sessionkey Aushandlung

gesetzt wurde, noch besteht (vergleiche (N030.700)), existieren diese Sessionkeys. Solange die Sessionkeys existieren, MUSS das COS sie erfolgreich finden.**[<=]**

9.2.3.1 Suche nach einem geheimen Schlüsselobjekt

In diesem Kapitel werden symmetrische Authentisierungsobjekte gemäß 8.6.1, symmetrische Kartenverbindungsobjekte gemäß 8.6.2 und private Schlüsselobjekte gemäß 8.6.3 gemeinsam behandelt, da nach ihnen auf gleiche Art und Weise gesucht wird. Sie werden im Rahmen diverser kryptographischer Operationen verwendet. Symmetrische Authentisierungsobjekte und symmetrische Kartenverbindungsobjekte werden zudem auch bei der Auswertung von Zugriffsregeln (siehe (N022.300)) verwendet. Bei dieser Art der Schlüsselsuche handelt es sich um eine karteninterne Funktionalität, bei der viele Implementierungsdetails eine Rolle spielen. An der Schnittstelle "Interpreter" (siehe CosA_e09) wird das im Folgenden festgelegte Verhalten sichtbar:

Tabelle 65: CosT_d11: Definition der Funktion SearchSecretkey(...)

Input:	*keyReference*	Gemäß (N099.600)
	algID	Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt, oder "Wildcard"
	startFolder	Ordner, bei dem die Suche nach einem Schlüsselobjekt startet
Output:	*key*	Enthält das gefundene Schlüsselobjekt
Errors:	*keyNotFound*	Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden.
	notSupported	Der Schlüssel unterstützt den von *algID* geforderten Algorithmus nicht.
Notation:		*key* = SearchSecretKey(*startFolder*, *keyReference*, *algID*)

G2_N021.100 - (N021.100) K_COS
Das COS MUSS den Inputparameter *keyReference* gemäß (N099.600) wie folgt in seine Bestandteile zerlegen:
a. *identifier* = *keyReference* mod 128
b. *location* = *keyReference* – *identifier***[<=]**

A_14136 - (N021.200) K_COS
Wenn *location* den Wert '00' = 0 besitzt und damit ein globales Schlüsselobjekt adressiert, dann MUSS das COS im Wurzelverzeichnis*root* des Objektsystems nach einem Schlüsselobjekt suchen, dessen Attribut *keyIdentifier* identisch zu *identifier* ist.**[<=]**

G2_N021.200.a - (N021.200)a K_COS
Wenn ein solches Schlüsselobjekt existiert,
dann MUSS das COS als Outputparameter *key* verwenden.**[<=]**

G2_N021.200.b - (N021.200)b K_COS
Wenn ein solches Schlüsselobjekt nicht existiert,
dann MUSS das COS den Algorithmus mit *keyNotFound* abbrechen.**[<=]**

G2_N021.300 - (N021.300) K_COS

Wenn *location* den Wert '80' = 128 besitzt und damit ein DF-spezifisches Schlüsselobjekt adressiert,
dann MUSS das COS die "lokale Variable"*folder* auf *startFolder* setzen.**[<=]**

G2_N021.400.a - (N021.400)a K_COS
Wenn *folder* auf das Wurzelverzeichnis *root* zeigt,
dann MUSS das COS den Algorithmus mit *keyNotFound* abbrechen.**[<=]**

A_14137 - (N021.400)b K_COS
Das COS MUSS in *folder* nach einem Schlüsselobjekt suchen, dessen Attribut *keyIdentifier* identisch zu *identifier* ist.**[<=]**

G2_N021.400.b.1 - (N021.400)b.1 K_COS
Wenn ein solches Schlüsselobjekt existiert,
dann MUSS das COS es als Outputparameter *key* verwenden.**[<=]**

G2_N021.400.b.2 - (N021.400)b.2 K_COS
Wenn ein solches Schlüsselobjekt nicht existiert,
dann MUSS das COS *folder* auf den nächsthöheren Ordner setzen und den Algorithmus mit Schritt (N021.400)a fortsetzen.**[<=]**

G2_N021.500.a - (N021.500)a K_COS
Wenn der Outputparameter *key* ein symmetrischer Schlüssel ist und dessen Attribut *algorithmIdentifier* nicht zum Parameter *algID* passt, dann MUSS das COS den Algorithmus mit *notSupported* abbrechen.**[<=]**

A_14138 - (N021.500)b K_COS
Wenn der Outputparameter *key* ein privater Schlüssel ist, dann MUSS das COS in den Elementen von *listAlgorithmIdentifier* nach einem Element suchen, dessen *seIdentifier* (siehe (N017.400)b.i) identisch zum Attribut *seIdentifier* (siehe (N030.000)a) des Ordners ist, in welchem *key* als child enthalten ist.**[<=]**

G2_N021.500.b.1 - (N021.500)b.1 K_COS
Wenn ein solches Element nicht existiert,
dann MUSS das COS den Algorithmus mit *notSupported* abbrechen.**[<=]**

G2_N021.500.b.2 - (N021.500)b.2 K_COS
Wenn ein solches Element existiert, aber die Menge *setAlgorithmIdentifier* (siehe (N017.400)b.2) das Element *algID* nicht enthält,
dann MUSS das COS den Algorithmus mit *notSupported* abbrechen.**[<=]**

9.2.3.2 Suche nach einem öffentlichen Schlüsselobjekt

Öffentliche Schlüsselobjekte werden zum Importieren von Schlüsseln mittels Zertifikaten und im Rahmen von Authentisierungsprotokollen oder Verschlüsselung verwendet. Die Unterklasse "Öffentliche Authentisierungsobjekte" wird zudem auch bei der Auswertung von Zugriffsregeln (siehe (N022.400)~~und (N022.500)~~) verwendet. Bei dieser Art der Schlüsselsuche handelt es sich um eine karteninterne Funktionalität, bei der viele Implementierungsdetails eine Rolle spielen. An der Schnittstelle "Interpreter" (siehe CosA_e09) wird das im Folgenden festgelegte Verhalten sichtbar:

Tabelle 66: CosT_995: Definition der Funktion SearchPublicKey(...)

Input:	*identifier*	Gemäß (N018.500), (N019.100)a, (N019.500)a, (N019.822)a
	algID	Kryptographisches Verfahren, welches der zu suchende

		Schlüssel unterstützt
	startFolder	Ordner, aus dem heraus die Suche startet
Output:	*key*	Enthält das gefundene Schlüsselobjekt
Errors:	*keyNotFound*	Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden.
	notSupported	Der Schlüssel unterstützt den von *algID* geforderten Algorithmus nicht.
Notation		*key* = SearchPublicKey(*startFolder*, *identifier*, *algID*)

Im Rahmen der Suche nach öffentlichen Schlüsselobjekten wird folgender Algorithmus durchgeführt:

A_14140 - (N021.600) K_COS
Das COS MUSS die "lokale Variable "*folder* auf *startFolder* setzen.**[<=]**

G2_N021.600.a - (N021.600)a K_COS
Das COS MUSS in *allPublicKeyList* nach einem Schlüsselobjekt suchen,
1. dessen Attribut *keyIdentifier* identisch zu *identifier* ist und
2. das dem Ordner *folder* zugeordnet ist.**[<=]**

G2_N021.600.b.1 - (N021.600)b.1 K_COS
Wenn ein solches Schlüsselobjekt existiert,
dann MUSS das COS es als Outputparameter *key* verwenden.**[<=]**

G2_N021.600.b.2.i - (N021.600)b.2.i K_COS
Wenn ein solches Schlüsselobjekt nicht existiert und *folder* ist gleich *root*,
dann MUSS das COS den Algorithmus mit *keyNotFound* abbrechen.**[<=]**

G2_N021.600.b.2.ii - (N021.600)b.2.ii K_COS
Wenn ein solches Schlüsselobjekt nicht existiert und *folder* ist ungleich *root*,
dann MUSS das COS *folder* auf den nächsthöheren Ordner setzen und der Algorithmus mit Schritt (N021.600)a fortsetzen.**[<=]**

G2_N021.605 - (N021.605) K_COS
Wenn das Attribut *oid* des Outputparameters *key* nicht äquivalent zur vorgegebenen *algID* ist (siehe (N021.610)), genau dann MUSS das COS den Algorithmus mit *notSupported* abbrechen.**[<=]**

(N021.610)a ist absichtlich leer.
(N021.610)b ist absichtlich leer.

G2_N021.610.a - (N021.610)a K_COS
~~Das COS MUSS *oid* = sigS_ISO9796-2Withrsa_sha256 (siehe CosT_a91) als äquivalent zum *algID* = verifyCertificate (siehe CosT_c40) betrachten.~~**[<=]**

G2_N021.610.b - (N021.610)b K_COS
~~Das COS MUSS *oid* = authS_ISO9796-2Withrsa_sha256_mutual (siehe CosT_a91) als äquivalent zum *algID* aus der folgenden Menge betrachten (siehe CosT_7a2): {rsaRoleCheck, rsaSessionkey4SM, rsaSessionkey4TC}.~~**[<=]**

G2_N021.610.c - (N021.610)c K_COS

Das COS MUSS einen *oid* aus der Menge {ecdsa-with-SHA256, ecdsa-with-SHA384, ecdsa-with-SHA512} (siehe CosT_a91) als äquivalent zum *algID* = verifyCertificate (siehe CosT_c40) betrachten.**[<=]**

G2_N021.610.d - (N021.610)d K_COS
Das COS MUSS einen *oid* aus der Menge {authS_gemSpec-COS-G2_ecc-with-sha256, authS_gemSpec-COS-G2_ecc-with-sha384, authS_gemSpec-COS-G2_ecc-with-sha512} (siehe CosT_a91) als äquivalent zum *algID* aus der folgenden Menge betrachten (siehe CosT_7a2): {elcAsynchronAdmin, elcRoleCheck, elcSessionkey4SM, elcSessionkey4TC}.**[<=]**

G2_N021.610.e - (N021.610)e K_COS
Das COS MUSS *oid* = id-RSAES-OAEP (siehe CosT_a91) als äquivalent zum *algID* = rsaEncipherOaep (siehe CosT_605) betrachten.**[<=]**

(N021.610)f Diese Anforderung ist absichtlich leer.

G2_N021.610.g - (N021.610)g K_COS
Das COS MUSS *oid* = id-ELC-shared-secret-calculation (siehe CosT_a91) als äquivalent zum *algID* = elcSharedSecretCalculation (siehe CosT_605) betrachten.**[<=]**

9.3 Cache für öffentliche Schlüsselobjekte

Die in diesem Unterkapitel beschriebene Funktion speichert ein übergebenes "öffentliches Signaturprüfobjekt" (siehe8.6.4.1) oder "öffentliches Authentisierungsobjekt" (siehe8.6.4.2) im Cache.

Tabelle 67: CosT_995: Definition der Funktion StoreInCache(...)

Input:	*puk*	"Öffentliches Signaturprüfobjekt" (siehe8.6.4.1) oder "Öffentliches Authentisierungsobjekt" (siehe8.6.4.2)
	folder	Ordner, dem *puk* zuzuordnen ist
Output:	*isPersistent*	True => *puk* wurde in *persistentCache* gespeichert False => *puk* wurde in *volatileCache* gespeichert
Notation		*isPersistent* = StoreInCache(*folder*, *puk*)

(N021.630) ist absichtlich leer.

G2_N021.630 - (N021.630)a K_COS, Option_RSA_CVC
~~Wenn *puk.publicKey* vom Typ *publicRsaKey* ist, dann MUSS das COS *puk* in Cache speichern.~~**[<=]**

A_15682 - (N021.630)b K_TST, Option_RSA_CVC
~~Wenn *puk.publicKey* vom Typ *publicRsaKey* ist, dann DARF einem Prüfling die funktionale Eignung NICHT mit der Begründung verwehrt werden *puk* würde in *persistentCache* gespeichert.~~**[<=]**

A_15683 - (N021.630)c K_TST, Option_RSA_CVC
~~Wenn *puk.publicKey* vom Typ *publicRsaKey* ist, dann DARF einem Prüfling die funktionale Eignung NICHT mit der Begründung verwehrt werden *puk* würde in *volatileCache* gespeichert.~~**[<=]**

G2_N021.633.a - (N021.633)a K_COS
Wenn *puk.publicKey* vom Typ *publicElcKey* ist, dann MUSS das COS *puk* in Cache speichern.**[<=]**

A_15684 - (N021.633)b.1 K_TST
Wenn *puk.publicKey* vom Typ *publicElcKey* ist und in Cache ist bereits ein Schlüsselobjekt vorhanden, dessen Attribut *keyIdentifier* identisch ist zu *puk.keyIdentifier* und dieses im Cache vorhandene Schlüsselobjekt unterscheidet sich hinsichtlich Schlüssellänge oder Schlüsseltyp (RSA oder ELC) von *puk*, dann MUSS im Rahmen der funktionalen Prüfung jedes COS-Verhalten akzeptiert werden.**[<=]**

G2_N021.633.b.2 - (N021.633)b.2 K_COS
Wenn *puk.publicKey* vom Typ *publicElcKey* ist und in Cache ist bereits ein Schlüsselobjekt vorhanden, dessen Attribut *keyIdentifier* identisch ist zu *puk.keyIdentifier* und dieses im Cache vorhandene Schlüsselobjekt ist hinsichtlich Schlüssellänge und Schlüsseltyp identisch zu *puk*, dann MUSS das COS das im Cache vorhandene Schlüsselobjekt mit identischem *keyIdentifier* durch *puk* ersetzen.**[<=]**

A_18255 - (N021.633)c K_COS
Wenn *puk.publicKey* vom Typ *publicElcKey* ist und in Cache ist kein Schlüsselobjekt vorhanden dessen Attribut *keyIdentifier* identisch ist zu *puk.keyIdentifier*, dann MUSS das COS wie folgt vorgehen:**[<=]**

G2_N021.633.c.1 - (N021.633)c.1 K_COS
Wenn in *persistentCache* nicht genügend Platz vorhanden ist, dann MUSS das COS gemäß (N021.636) freien Platz schaffen.**[<=]**

G2_N021.633.c.2.i - (N021.633)c.2.i K_COS
Wenn in *persistentCache* genügend Platz vorhanden ist, dann MUSS das COS *puk* in *persistentCache* speichern.**[<=]**

G2_N021.633.c.2.ii - (N021.633)c.2.ii K_COS
Wenn in *persistentCache* nicht genügend Platz vorhanden ist, obwohl gemäß (N021.633)c.1 versucht wurde Platz zu schaffen, dann MUSS das COS *puk* in *volatileCache* speichern.**[<=]**

G2_N021.636 - (N021.636) K_COS
Das COS MUSS anhand des folgenden Algorithmus versuchen freien Platz in *peristentCache* zu schaffen:

a. Schritt 1: In *persistentCache* wird nach abgelaufenen EE-Schlüssel(n) und abgelaufenen CA-Schlüssel(n) gesucht, unabhängig davon, ob diese in irgendeinem logischen Kanal in *channelContext*. *keyReferenceList* eingetragen sind oder nicht. Wenn solche Schlüsselobjekte
   i. existieren, dann wird eines oder mehrere dieser Schlüsselobjekte aus *persistentCache* entfernt. Wenn dadurch genügend Platz geschaffen wird, dann bricht dieser Algorithmus ab, sonst wird dieser Schritt wiederholt.
   ii. nicht existieren, dann wird mit Schritt 2 fortgefahren.
b. Schritt 2: Dieser Schritt wird erforderlich, wenn durch das Löschen abgelaufener EE-Schlüssel oder abgelaufener CA-Schlüssel nicht genügend Platz geschaffen wurde. Dann wird in *persistentCache* nach EE-Schlüssel(n) oder CA-Schlüssel(n) gesucht, die in keinem logischen Kanal in *channelContext*. *keyReferenceList* eingetragen sind. Wenn solche Schlüsselobjekte

i. existieren, dann wird eines dieser Schlüsselobjekte aus *persistentCache* entfernt. Wenn dadurch genügend Platz geschaffen wird, bricht dieser Algorithmus ab, sonst wird dieser Schritt wiederholt.

ii. nicht existieren, dann wird mit Schritt 3 fortgefahren.

c. Schritt 3: Dieser Schritt wird erforderlich, wenn *persistentCache* voll ist mit Sicherheitsankern und Schlüsseln, die in *channelContext*. *keyReferenceList* selektiert sind. Wenn in *volatileCache*

i. genügend freier Platz vorhanden ist, dann bricht dieser Algorithmus ab.

ii. nicht genügend Platz vorhanden, dann MUSS ein beliebiges Schlüsselobjekt aus Cache entfernt und dieser Schritt wiederholt werden.

[<=]

(N021.638)a Diese Anforderung ist absichtlich leer.

(N021.638)b Diese Anforderung ist absichtlich leer.

G2_N021.638.a - (N021.638)a K_COS
~~Wird ein öffentliches Schlüsselobjekt aus Cache entfernt, so DARF das COS in KEINEM logischen Kanal den Inhalt von *globalSecurityList* (siehe (N029.900)e) ändern.~~**[<=]**

G2_N021.638.b - (N021.638)b K_COS
~~Wird ein öffentliches Schlüsselobjekt aus Cache entfernt, so DARF das COS in KEINEM logischen Kanal den Inhalt von *dfSpecificSecurityList* (siehe (N029.900)f) ändern.~~**[<=]**

G2_N021.638.c - (N021.638)c K_COS
Wird ein öffentliches Schlüsselobjekt aus Cache entfernt, so DARF das COS in KEINEM logischen Kanal den Inhalt von *bitSecurityList* (siehe (N029.900)h) ändern.**[<=]**

A_15685 - (N021.638)d K_TST
Wenn ein öffentliches Schlüsselobjekt aus Cache entfernt wird, dann MUSS es für die funktionale Eignung zulässig sein, dass das COS in beliebigen logischen Kanälen Einträge in *keyReferenceList*, welche eine Referenz auf das entfernte Schlüsselobjekt enthalten,
1. verändert oder
2. unverändert lässt. **[<=]**

A_15686 - (N021.639) K_COS
Für den Rückgabewert gilt: Wenn *puk* in *persistentCache* gespeichert wurde,
dann MUSS das COS als Rückgabewert *persistent* = True wählen,
sonst False.**[<=]**

Hinweis CosH_df6: Die Bedingungen, die zu (N021.633)b.1 führen, sind nur erfüllt, wenn es zwei verschiedene Instanzen mit derselben Schlüsselreferenz gibt. Das wird in [gemSpec_PKI] ausgeschlossen.

Hinweis CosH_d42: In (N021.636) ist eine Strategie beschrieben, die in mehreren Schritten versucht ein "unwichtiges" Schlüsselobjekt im Cache zu finden. Wenn so ein "unwichtiges" Schlüsselobjekt existiert, dann wird es aus dem Cache entfernt mit dem Ziel Platz für puk *zu schaffen. Der Text wurde nicht mit dem Ziel geschrieben eine bestimmte Implementierung vorzugeben, sondern mit dem Ziel das gewünschte Verhalten zu verdeutlichen. Damit logische Kanäle bezüglich importierter Schlüssel unabhängig voneinander sind, ist* persistentCache *so zu dimensionieren, dass Schritt 3 in (N021.636)c nicht erforderlich wird. Dem COS ist es nicht möglich dies sicherzustellen. Beispielsweise ist es der externen Welt möglich sehr viele Sicherheitsanker zu importieren. In praxisrelevanten Fällen wären dann alte, überflüssige Sicherheitsanker im Rahmen einer Kartenadministration zu löschen um für freien Platz in*

persistentCache *zu sorgen. Nach Abarbeitung des Algorithmus in (N021.636) ist im Cache genügend Platz für* puk *frei.*

Hinweis CosH_b42: Wenn die Bedingung in (N021.636)a auf mehrere Schlüsselobjekte zutrifft, so wird hier weder festgelegt welches gelöscht wird, noch ob nur eines oder alle gelöscht werden.

Hinweis CosH_0c3: Wenn die Bedingung in (N021.636)a auf mehrere zutrifft aber nur eines gelöscht wird, so erscheint es sinnvoll, das mit dem kleinsten CXD (am längsten abgelaufen) zu entfernen.

Hinweis CosH_b54: Wenn die Bedingung (N021.636)b auf mehrere Schlüsselobjekte zutrifft, so wird nicht festgelegt welches gelöscht wird. Es erscheint sinnvoll, aus der Menge der EE-Schlüssel das mit dem kleinsten CXD zu entfernen.

10 Zugriffskontrolle (normativ)

Fast alle in Kapitel 14 beschriebenen Kommandos werden durch Zugriffsregeln geschützt. Das bedeutet, dass das Betriebssystem kontrolliert, ob der Sicherheitsstatus für die Ausführung der Operation ausreichend ist. Die in diesem Kapitel aufgeführten Regeln bilden eine Untermenge der in [ISO/IEC 7816-4#9.3.3] Expanded format definierten Regeln.

10.1 Zugriffsart

Die Zugriffsart (access mode) zeigt an, ob die der Zugriffsart zugeordnete Zugriffsbedingung im Rahmen einer Zugriffsregelprüfung auszuwerten ist. Bei der Spezifikation von Anwendungen sind folgende Regeln einzuhalten:

G2_N021.700 - (N021.700) K_Anwendungsspezifikation {K_Karte}
Eine Zugriffsart MUSS eine Liste von Kommandobeschreibungen sein.**[<=]**

Als Untermenge zu [ISO/IEC 7816-4#Tabelle 32] gilt für jedes Listenelement folgendes:

A_15687 - (N021.800)a K_Anwendungsspezifikation {K_Karte}
Ein Listenelement MUSS den Namen eines Kommandos enthalten, der gemäß Kapitel 14 äquivalent ist zu genau einer Kombination aus CLA Byte (siehe 11.5.1) und INS Byte (siehe 11.5.2).**[<=]**

G2_N021.800.a.1 - (N021.800)a.1 K_Anwendungsspezifikation {K_Karte}
Das CLA Byte in einer Zugriffsart MUSS die Kanalnummer null enthalten. Das ist so zu interpretieren, dass die Zugriffsart für beliebige Kanalnummern gilt.**[<=]**

G2_N021.800.a.2 - (N021.800)a.2 K_Anwendungsspezifikation {K_Karte}
Das CLA Byte in einer Zugriffsart DARF KEIN Secure Messaging anzeigen. Das ist so zu interpretieren, dass die Zugriffsart sowohl für ungeschützte Nachrichtenübertragung, als auch für beliebig per Secure Messaging geschützte Nachrichtenübertragung gilt.**[<=]**

Innerhalb eines Listenelementes ist der Parameter P1 (siehe 11.5.3) optional. Das bedeutet:

G2_N021.800.b.1 - (N021.800)b.1 K_Anwendungsspezifikation {K_Karte}
Es MUSS möglich sein, dass ein Listenelement genau einen Wert für den Parameter P1 enthält. Das ist so zu interpretieren, dass die Zugriffsart nur für diesen Wert von P1 gilt.**[<=]**

G2_N021.800.b.2 - (N021.800)b.2 K_Anwendungsspezifikation {K_Karte}
Es MUSS möglich sein, dass ein Listenelement keinen Parameter P1 enthält. Das ist so zu interpretieren, dass die Zugriffsart für beliebige Werte von P1 gilt.**[<=]**

Innerhalb eines Listenelementes ist der Parameter P2 (siehe 11.5.4) optional. Das bedeutet:

G2_N021.800.c.1 - (N021.800)c.1 K_Anwendungsspezifikation {K_Karte}
Es MUSS möglich sein, dass ein Listenelement genau einen Wert für den Parameter P2 enthält. Das ist so zu interpretieren, dass die Zugriffsart nur für diesen Wert von P2 gilt.**[<=]**

G2_N021.800.c.2 - (N021.800)c.2 K_Anwendungsspezifikation {K_Karte}
Es MUSS möglich sein, dass ein Listenelement keinen Parameter P2 enthält. Das ist so zu interpretieren, dass die Zugriffsart für beliebige Werte von P2 gilt.**[<=]**

A_15690 - (N021.800)d K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Arten von Kommandobeschreibungen
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N021.900 - (N021.900) K_COS
Das COS MUSS bei der Auswertung der Zugriffsart diese genau dann als passend zur aktuellen Kommando-APDU ansehen, wenn es in der Liste der Kommandobeschreibungen ein Listenelement gibt, dessen Bestandteile identisch sind zu den Bestandteilen der Kommando-APDU an der Schnittstelle "Interpreter" (siehe CosA_e09).**[<=]**

10.2 Zugriffsbedingung

Eine Zugriffsbedingung ist ein boolescher Ausdruck. Bei der Spezifikation einer Anwendung sind folgende Regeln einzuhalten, wobei zur Beschreibung folgende Definitionen eingeführt werden:

Tabelle 68: CosT_6f1: Definition von Begriffen, die bei der Beschreibung von Zugriffsregeln verwendet werden

objectFolder	Wenn diese Zugriffsbedingung zu einem Objekt 1. vom Typ Ordner gehört, dann ist *objectFolder* gleich dieser Ordner 2. vom Typ Sessionkey gehört, dann ist *objectFolder* gleich dem Ordner, auf den *SessionkeyContext*. *folderSessionkeys* zeigt. 3. anderen Typs gehört, dann ist *objectFolder* gleich dem Ordner, dessen Attribut *children* dieses Objekt enthält
path(*folder*)	Pfad eines Ordners mit Namen *folder*. Zum Pfad gehört nach dieser Definition sowohl der Ordner *folder*, als auch alle seine übergeordneten Ebenen gemäß 9.1 einschließlich *root*

G2_N022.000 - (N022.000) K_COS
Das COS MUSS für das boolesche Elemente ALWAYS den Wert True verwenden.**[<=]**

G2_N022.100 - (N022.100) K_COS
Das COS MUSS für das boolesche Element NEVER den Wert False verwenden.**[<=]**

A_15706 - (N022.200) K_COS
Das COS MUSS für das boolesche Element PWD(*passwordReference*) **[<=]**

A_15707 - (N022.000)a
genau dann den Wert True verwenden, wenn die Passwortsuche gemäß CosT_65c mit den Inputparametern SearchPwd(*objectFolder, passwordReference*) ein Passwort findet und wenigstens eine der beiden folgenden Bedingungen erfüllt ist:**[<=]**

G2_N022.200.a.1 - (N022.200)a.1
Dieses Passwortobjekt ist entweder in *globalPasswordList* (siehe (N029.900)i) oder in *dfSpecificPasswordList* (siehe (N029.900)j) enthalten und das Attribut *securityStatusEvaluationCounter* dieses Passwortes ist ungleich null. Anschließend MUSS das COS das Attribut *securityStatusEvaluationCounter* dieses Passwortes um eins dekrementieren.**[<=]**

G2_N022.200.a.2 - (N022.200)a.2
Das Attribut *flagEnabled* dieses Passwortes besitzt den Wert False.**[<=]**

G2_N022.200.b - (N022.200)b
in allen anderen Fällenden Wert False verwenden.**[<=]**

A_15708 - (N022.210) K_TST
Wenn durch das Dekrementieren gemäß (N022.200)a.1 der Wert auf Null gefallen ist, dann MUSS es für die funktionale Eignung zulässig sein, dass das COS den Eintrag für dieses Passwort in einer der Listen *globalPasswordList* oder *dfSpecificPasswordList*
a. lässt, oder
b. entfernt.**[<=]**

A_15709 - (N022.300) K_COS
Das COS MUSS für das boolesche Element AUT(*keyReference*) **[<=]**

G2_N022.300.a - (N022.300)a
genau dann den Wert True verwenden, wenn die Schlüsselsuche gemäß CosT_d11 mit den Inputparametern SearchSecretKey (*objectFolder, keyReference*, "Wildcard") ein Schlüsselobjekt *key* findet und genau dieses Schlüsselobjekt entweder in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist.**[<=]**

G2_N022.300.b - (N022.300)b
in allen anderen Fällenden Wert False verwenden.**[<=]**

A_15710 - (N022.400) K_COS
Das COS MUSS für das boolesche Element AUT(CHAT)**[<=]**

G2_N022.400.a - (N022.400)a
genau dann Wert True verwenden, wenn es mindestens ein Element in *bitSecurityList* (siehe (N029.900)h) gibt, in welchem die OID übereinstimmt und mindestens dieselben Bits gesetzt sind wie in *accessFlags* (siehe auch CosH_1ea) und dieses Element ist einem Ordner in path(*objectFolder*) zugeordnet (siehe (N030.400)b).**[<=]**

G2_N022.400.b - (N022.400)b
in allen anderen Fällenden Wert False verwenden.**[<=]**

(N022.500) ist absichtlich leer.

A_15711 - (N022.500) K_COS, Option_RSA_CVC
~~Das COS MUSS für das boolesche Element AUT(*CHA*)~~**[<=]**

G2_N022.500.a - (N022.500)a
~~genau dann Wert True verwenden, wenn *CHA* in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist und *CHA* einem Ordner in path(*objectFolder*) zugeordnet ist (siehe (N030.300)b.2).~~**[<=]**

G2_N022.500.b - (N022.500)b
~~in allen anderen Fällenden Wert False verwenden.~~**[<=]**

Secure Messaging gemäß Kapitel 13 wird wie folgt in Zugriffsbedingungen verwendet:

G2_N022.600.a - (N022.600)a K_COS
Das COS MUSS für das boolesche Element SmMac(*keyInformation*) genau dann den Wert True verwenden, falls *SessionkeyContext.flagSessionEnabled* gleich SK4SM ist, und *SessionkeyContext. negotiationKeyInformation*
1. eine Schlüsselreferenz gemäß (N099.600) enthält und AUT(*keyInformation*) gemäß (N022.300) liefert True, oder
2. enthält *accessRights* gemäß (N019.700) mit
   a. dieser Punkt ist absichtlich leer.~~Option_RSA_CVC, *CHA* gemäß 7.1.1.4 und AUT(*keyInformation*) gemäß (N022.500) liefert True, oder~~
   b. CHAT und AUT(*keyInformation*) gemäß (N022.400) liefert True,
3. sonst False.**[<=]**

G2_N022.600.b - (N022.600)b K_COS
Das COS MUSS für das boolesche Element SmCmdEnc den Wert von *SessionkeyContext.flagCmdEnc* verwenden.**[<=]**

G2_N022.600.c - (N022.600)c K_COS
Das COS MUSS für das boolesche Element SmRspEnc stets den Wert True verwenden.**[<=]**

G2_N022.600.d - (N022.600)d K_COS
Das COS MUSS *SessionkeyContext.flagRspEnc* genau dann auf True setzen, wenn die Zugriffsbedingung das Element SmRspEnc enthält.**[<=]**

Die zuvor definierten booleschen Elemente lassen sich zu einem booleschen Ausdruck verbinden.

G2_N022.700 - (N022.700) K_COS
a. Das COS MUSS den AND-Operator unterstützen.
b. Das COS MUSS den OR-Operator unterstützen.**[<=]**

A_15712 - (N022.700)c K_Anwendungsspezifikation {K_Karte}
Eine Anwendungsspezifikation DARF in Zugriffsbedingungen neben AND- und OR-Operatoren KEINE weiteren Operatoren enthalten.**[<=]**

Eine Verschlüsselung ist nur bei gleichzeitiger MAC-Sicherung sinnvoll, das heißt:

G2_N022.710.a - (N022.710)a K_Anwendungsspezifikation {K_Karte}
Jedes boolsche Element SmCmdEnc MUSS durch den AND-Operator mit SmMac(*keyInformation*) verknüpft werden.**[<=]**

G2_N022.710.b - (N022.710)b K_Anwendungsspezifikation {K_Karte}
Jedes boolsche Element SmRspEnc MUSS durch den AND-Operator mit SmMac(*keyInformation*) verknüpft werden.**[<=]**

G2_N022.800 - (N022.800) K_COS
Das COS MUSS als Wert der Zugriffsbedingung das Ergebnis des booleschen Ausdrucks verwenden.**[<=]**

Für die Kommandos GET CHALLENGE (siehe (N098.900)), MANAGE CHANNEL (siehe (N099.545)), MANAGE SECURITY ENVIRONMENT (siehe (N104.100) und SELECT (siehe (N047.600)) ist nicht festgelegt, ob Zugriffsregeln auszuwerten sind oder nicht. Darum gilt für die Zugriffsbedingung dieser Kommandos:

G2_N022.810.a - (N022.810)a K_COS

Wenn ein COS für eines der Kommandos aus der Menge {GET CHALLENGE, MANAGE CHANNEL, MANAGE SECURITY ENVIRONMENT, SELECT} keine Zugriffsregeln auswertet, dann MUSS sich das COS so verhalten, als ob die Zugriffsbedingung für derartige Zugriffsarten ALWAYS lautet.**[<=]**

A_15714 - (N022.810)b K_Karte
Wenn ein COS für eines der Kommandos aus der Menge {GET CHALLENGE, MANAGE CHANNEL, MANAGE SECURITY ENVIRONMENT, SELECT} Zugriffsregeln auswertet, dann MUSS eine Karte so konfiguriert werden, dass**[<=]**

G2_N022.810.b.1 - (N022.810)b.1 K_Karte
für die kontaktbehaftete Kommunikation die Zugriffsbedingung des Kommandos ALWAYS lautet.**[<=]**

G2_N022.810.b.2.i - (N022.810)b.2.i K_Karte
für die kontaktlose Kommunikation MUSS die Zugriffsbedingung des Kommandos ALWAYS lauten, falls beabsichtigt ist, diese Schnittstelle zu nutzen.**[<=]**

G2_N022.810.b.2.ii - (N022.810)b.2.ii K_Karte
für die kontaktlose Kommunikation MUSS die Zugriffsbedingung des Kommandos NEVER lauten, falls beabsichtigt ist, diese Schnittstelle nicht zu nutzen.**[<=]**

Hinweis CosH_34d: Die normativen Regeln dieses Dokumentes, insbesondere die
– zu Attributen eines Passwortes (siehe 8.4 und 8.5)
– zu Statusänderungen bei erfolgreicher Benutzerverifikation (siehe (N082.900))
– zum booleschen Element PWD (siehe (N022.200))
sind so aufgebaut, dass nach erfolgreicher Benutzerverifikation
*– genau eine Operation (etwa Signatur) möglich ist (*startSSEC *= 1, siehe (N015.800)c.2), oder*
– genau n *Operationen möglich sind (*startSSEC = n > 1*), oder*
*– beliebig viele Operationen möglich sind (*startSSEC = "unendlich"*).*

Hinweis CosH_1ea: Obwohl in (N029.900)h.1 festgelegt ist, dass eine Listenlänge von eins für das COS hinreichend ist um spezifikationskonform zu sein, wird in den folgenden Beispielen mit einer Listenlänge von zwei gearbeitet um den Inhalt von (N022.400) zu erläutern. Zur Vereinfachung sind in den Beispielen OID nicht dargestellt.

a. *Für* bitSecurityList *= {0001, 0110} ist AUT(0011) gleich False, weil es in* bitSecurityList *kein Element gibt, in welchem die beiden letzten Bits gesetzt sind. Ein verODERn der Listen in* bitSecurityList *findet demnach nicht statt.*

b. *Für* bitSecurityList *= {0001, 1011} ist AUT(0011) gleich True, weil in* bitSecurityList *im zweiten Element ebenfalls die beiden letzten Bits gesetzt sind.*

Hinweis CosH_7fe: Gemäß (N022.600)a lässt sich für SmMac *festlegen mit welchem Schlüssel Secure Messaging erfolgt. Eine entsprechende Festlegung für* SmCmdEnc *oder* SmRspEnc*ist nicht erforderlich, da ausschließlich Sessionkeys betrachtet werden, wo sowohl* Kmac*als auch* Kenc *gemeinsam vereinbart werden.*

10.3 Zugriffsregel

Eine Zugriffsregel kombiniert Zugriffsart und Zugriffsbedingung und liefert bei der Auswertung ein boolesches Ergebnis. Bei der Spezifikation einer Anwendung sind folgende Regeln einzuhalten:

G2_N022.900 - (N022.900) K_Anwendungsspezifikation {K_Karte}
Eine Zugriffsregel MUSS eine Liste mit mindestens einem Element sein.**[<=]**

G2_N023.000 - (N023.000) K_Anwendungsspezifikation {K_Karte}
Ein Listenelement MUSS genau eine Zugriffsart gemäß 10.1 und eine Zugriffsbedingung gemäß 10.2 enthalten. **[<=]**

A_15715 - (N023.100) K_COS
Das COS MUSS für den booleschen Wert eines Listenelementes**[<=]**

G2_N023.100.a - (N023.100)a
genau dann den Wert True verwenden, wenn
1. die Zugriffsart gemäß (N021.900) passt und
2. der boolesche Wert der Zugriffsbedingung den Wert True hat.**[<=]**

G2_N023.100.b - (N023.100)b
in allen anderen Fällen den Wert False liefern (implizites NEVER).**[<=]**

A_15716 - (N023.200) K_COS
Das COS MUSS eine Zugriffsregel**[<=]**

G2_N023.200.a - (N023.200)a
genau dann als erfüllt ansehen, wenn mindestens ein Listenelement den Wert True liefert.**[<=]**

G2_N023.200.b - (N023.200)b
in allen anderen Fällen als nicht erfüllt ansehen.**[<=]**

A_15717 - (N023.300) K_Anwendungsspezifikation {K_Karte}
Eine Zugriffsregel, welche die übrigen Bedingungen in Kapitel 10 erfüllt, DARF bei Codierung gemäß [ISO/IEC 7816-4#9.3.3] NICHT mehr als 240 Oktette beanspruchen.**[<=]**

10.4 Zugriffsregelauswertung

Dieses Unterkapitel beschreibt wie das COS feststellt, ob die Ausführung eines bestimmten Kommandos erlaubt oder verboten ist.

Tabelle 69: CosT_fe2: Definition der Funktion AccessRuleEvaluation(...)

Input:	*obj*	Instanz eines Objektes vom Typ Ordner, Datei, Passwortobjekt oder Schlüsselobjekt
	CLA *INS* *P1* *P2*	CLA Byte INS Byte Parameter P1 Parameter P2
Output:	*b*	Boolean, True bedeutet: Kommandoausführung erlaubt; False bedeutet: Kommandoausführung verboten
Errors:	–	Keine
Notation:		*b* = AccessRuleEvaluation(*obj*, *CLA*, *INS*, *P1*, *P2*)

Schritt 1: Handelt es sich bei *obj*

G2_N023.400.a - (N023.400)a K_COS

um einen Ordner, dann MUSS *se* = *seIdentifier* dieses Ordners sein (siehe (N030.000)a).**[<=]**

G2_N023.400.b - (N023.400)b K_COS
um einen Sessionkey, dann MUSS *se* = *seIdentifier* des Ordners sein (siehe (N030.000)a), auf den *SessionkeyContext. folderSessionkeys* zeigt.**[<=]**

G2_N023.400.c - (N023.400)c K_COS
sonst MUSS *se* = *seIdentifier* des Ordners sein (siehe (N030.000)a), der *obj* in seinem Attribut *children* (siehe (N010.000)) enthält.**[<=]**

A_15718 - (N023.500) K_COS
Schritt 2: Das COS MUSS unter Berücksichtigung des aktuell aktiven Übertragungsprotokolls (kontaktbehaftet oder kontaktlos) des logischen Wertes von *obj.lifeCycleStatus* (siehe (N020.600)) und *se* aus *obj. accessRules*, welches vom Typ *interfaceDependentAccessRules* ist, die passende Zugriffsregel auswählen (siehe 8.1.4).**[<=]**

A_15719 - (N023.600) K_COS
Schritt 3: Das COS MUSS anhand von *CLA*, *INS*, *P1* und *P2* ermitteln, ob die in (N023.500) ausgewählte Zugriffsregel erfüllt ist (siehe 10.3).**[<=]**

Schritt 4: Ist die ausgewählte Zugriffsregel

G2_N023.700.a - (N023.700)a K_COS
erfüllt, dann MUSS *b* = True gelten.**[<=]**

G2_N023.700.b - (N023.700)b K_COS
sonst MUSS *b* = False gelten.**[<=]**

Hinweis CosH_c0a: Wenn in folgender, beispielhafter Zugriffsbedingung (PWD1 OR PWD2) beide Passwortobjekte mit 0 < SSEC < unendlich vorkommen, dann ist es herstellerspezifisch, ob beide SSEC oder nur einer und in diesem Falle welcher dekrementiert wird.

(N023.740) ist absichtlich leer.

10.5 Auslesen von Zugriffsregeln

~~In 10.1 bis 10.4 wird festgelegt in welchem Rahmen sich eine Anwendungsspezifikation im Bereich Zugriffsregeln zu bewegen hat. Zur COS-internen Codierung von Zugriffsregeln gibt es keine Anforderung. Die COS-interne Codierung von Zugriffsregeln ist und bleibt herstellerspezifisch und liegt damit außerhalb des Regelungsbereiches dieses Dokumentes.~~

~~Das Konzept zur Evaluierung und damit zur Zulassung des COS sieht ab der Generation 2 vor, dass Zugriffsregeln aus einer Smartcard auslesbar sind. Dieses Unterkapitel nennt die zugehörigen Anforderungen:~~

~~*Hinweis CosH_89e: Die folgenden Anforderungen beziehen sich ausschließlich auf Zugriffsregeln, die Teil einer Anwendungsspezifikation sind. Da alle Anwendungsspezifikationen auf den Webseiten der gematik allgemein zugänglich veröffentlicht werden, sind sie öffentlich. Aus datenschutzrechtlicher Sicht ist deshalb gegen das Auslesen dieser Zugriffsregeln nichts einzuwenden. Das Auslesen dient dem Nachweis, dass eine Smartcard Zugriffsregeln semantisch spezifikationskonform umsetzt.*~~

A_15720 - (N023.740)a.1 K_Karte
~~Der Hersteller MUSS eine Karte so konfigurieren, dass jede Zugriffsregel, die in einer Anwendungspezifikation definiert ist, auslesbar ist.~~**[<=]**

A_15721 - (N023.740)a.2 K_Karte
~~Der Hersteller SOLL eine Karte so konfigurieren, dass jede Zugriffsregel, die in einer Anwendungspezifikation definiert ist, frei auslesbar ist, das heißt die Zugriffsregel zum Auslesen von Zugriffsregeln SOLL "read ALWAYS" sein.~~**[<=]**

A_15722 - (N023.740)b K_TST
~~Für die funktionale Eignung MUSS es zulässig sein, dass Zugriffsregeln für Objekte, die nicht in einer Anwendungsspezifikation definiert sind (etwa herstellerspezifische zusätzliche Objekte, EF.Pwd, etc.)~~
~~1. auslesbar oder~~
~~2. nicht auslesbar~~
~~sind.~~**[<=]**

11 Kommunikation (normativ)

11.1 Request - Response

Für Smartcards entspricht es dem Stand der Technik, dass sie Nachrichten mit einem externen Kommunikationspartner über einen Kanal im Halbduplex-Verfahren austauschen. Zudem arbeiten sie ähnlich wie ein Server. Das bedeutet, dass der externe Kommunikationspartner eine Nachricht (Kommando) über den Kanal schickt. Diese Nachricht (Kommando) wird von der Smartcard verarbeitet. Anschließend sendet die Smartcard über denselben Kanal eine Nachricht (Antwort) zurück. Erst nach dem vollständigen Empfang der Smartcard-Nachricht hat der externe Kommunikationspartner die Möglichkeit, eine weitere Nachricht zu schicken.

11.2 Elektrische Schnittstellen

Dieses Kapitel behandelt elektrische Schnittstellen zum COS. Das Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11] ist verpflichtend und wird in 11.2.1 näher spezifiziert. Eine Datenübertragung gemäß [ISO/IEC 7816-12] (siehe 11.2.2) oder kontaktlos gemäß ISO/IEC 14443 (siehe 11.2.3) ist optional.

11.2.1 Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11]

Dieses Kapitel behandelt das kontaktbehaftete Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11]. Die Unterstützung dieses Übertragungsprotokolls ist verpflichtend.

(N023.800) Diese Anforderung ist absichtlich leer.

G2_N023.801 - (N023.801) K_COS
Das COS MUSS das kontaktbehaftete Übertragungsprotokoll T=1 unterstützen. Das Intervall [(N023.820), (N024.220)] spezifiziert die diesbezüglichen Anforderungen.**[<=]**

G2_N023.820.a - (N023.820)a K_IC
Das IC einer Smartcard MUSS elektrische Eigenschaften gemäß [EMV®_Book-1#5.3] aufweisen.**[<=]**

G2_N023.820.b - (N023.820)b K_IC
Das IC einer Smartcard MUSS die Spannungklasse "Class A" unterstützen.**[<=]**

G2_N023.820.c - (N023.820)c K_IC
Das IC einer Smartcard MUSS die Spannungsklasse "Class B" unterstützen.**[<=]**

A_15723 - (N023.820)d K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das IC einer Smartcard die Spannungsklasse "Class C" unterstützt.**[<=]**

A_15724 - (N023.820)e K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das IC einer Smartcard die Spannungsklasse "Class C" nicht unterstützt.**[<=]**

G2_N023.820.f - (N023.820)f K_Karte
Der Hersteller einer Karte MUSS diese so konfigurieren, dass alle Spannungsklassen, welche die Smartcard im ATR oder in einem EF.ATR anzeigt, konform zu [EMV®_Book-1#5.3] implementiert sind (siehe auch (N024.100)i).**[<=]**

(N023.900) Diese Anforderung ist absichtlich leer.

Eine Kartensession gliedert sich in die Abschnitte Kartenaktivierung, Reset, Nachrichtenaustausch und Deaktivierung. Dabei gilt:

G2_N023.920.a - (N023.920)a K_externeWelt {K_Karte}
Der externe Kommunikationspartner MUSS die Smartcard gemäß [EMV®_Book-1#6.1.2] aktivieren.**[<=]**

G2_N023.920.b - (N023.920)b K_externeWelt {K_Karte}
Im Anschluss an die Aktivierung gemäß (N023.920)a MUSS der externe Kommunikationspartner einen Cold Reset gemäß [EMV®_Book-1#6.1.3.1] ausführen.**[<=]**

G2_N023.920.c - (N023.920)c K_IC
Das IC einer Smartcard MUSS einen Warm Reset gemäß [EMV®_Book-1#6.1.3.2] unterstützen.**[<=]**

G2_N023.920.d - (N023.920)d K_externeWelt {K_Karte}
Der externe Kommunikationspartner MUSS die "Information Field Size" IFSD durch einen IFSD-Request (siehe [ISO/IEC 7816-3#11.4.2]) auf den Wert 'FE' = 254 setzen.**[<=]**

(N023.920)e Dieser Punkt ist absichtlich leer.
Der externe Kommunikationspartner tauscht nach Abschluss eines Resets mittels des hier spezifizierten Übertragungsprotokolls beliebige Nachrichten aus.

G2_N023.920.f.1 - (N023.920)f.1 K_externeWelt {K_Karte}
Der externe Kommunikationspartner SOLL das IC einer Smartcard gemäß [EMV®_Book-1#6.1.5] deaktivieren.**[<=]**

A_15725 - (N023.920)f.2 K_COS
~~Die Deaktivierung KANN auf andere Art erfolgen.~~ Das COS MUSS so robust sein, dass eine Deaktivierung möglich ist, die von den Vorgaben aus [EMV®_Book-1#6.1.5] abweicht (beispielsweise "Karte ziehen").**[<=]**

G2_N023.920.f.3 - (N023.920)f.3 K_COS
Geschieht die Deaktivierung zu einem Zeitpunkt, in welchem eine transaktionsgeschützte Schreiboperation durchgeführt wird (siehe 14.1), so MUSS das COS dafür Sorge tragen, dass gemäß den Regeln in (N034.600) bzw. (N034.700) verfahren wird, bevor die nächste Kommando-APDU von der Komponente "Cmd Interpreter" in CosA_e09 bearbeitet wird.**[<=]**

G2_N023.940 - (N023.940) K_COS
Wenn das COS keinen IFSD-Request empfängt (siehe (N023.920)d),
dann MUSS das COS mit einem Wert IFSD=254 (siehe [EMV®_Book-1#9.2.4.2.1]) arbeiten.**[<=]**

(N024.000) Diese Anforderung ist absichtlich leer.

G2_N024.020 - (N024.020) K_IC, K_externeWelt {K_Karte}
Im Rahmen des hier spezifizierten Übertragungsprotokolls MÜSSEN auf der physikalischen Schnittstelle (siehe CosA_e09) Character gemäß [EMV®_Book-1#7.2] verwendet werden.**[<=]**

Ein Cold Reset gemäß (N023.920)b und ein Warm Reset gemäß (N023.920)c lassen sich als spezielle "Nachrichten" des externen Kommunikationspartners auffassen, die von der Smartcard mit einem Answer to Reset (ATR) gemäß [ISO/IEC 7816-3#8.2] beantwortet wird. Dabei gilt:

G2_N024.100.a - (N024.100)a K_COS
Das COS MUSS den ATR gemäß [EMV®_Book-1#8.1] versenden.**[<=]**

G2_N024.100.b - (N024.100)b K_COS
Das COS MUSS als "initial character" im ATR den Wert '3B' verwenden. Daraus ergibt sich, dass die Kommunikation mittels "direct convention" (siehe [ISO/IEC 7816-3#8.1]) erfolgt.**[<=]**

G2_N024.100.c - (N024.100)c K_COS
Das COS MUSS im Rahmen eines Cold Reset für das TA1 Byte im ATR einen Wert aus der Menge {'18', '95', '96', '97'} verwenden.**[<=]**

G2_N024.100.d - (N024.100)d K_COS
Das COS MUSS im Rahmen eines Warm Reset für das TA1 Byte im ATR einen Wert aus der Menge {'18', '95', '96'} verwenden.**[<=]**

G2_N024.100.e.1 - (N024.100)e.1 K_COS
Das COS SOLL im ATR ein TC1 Byte enthalten.**[<=]**

A_15726 - (N024.100)e.2 K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS auf ein TC1 Byte in einem ATR verzichtet.**[<=]**

G2_N024.100.e.3 - (N024.100)e.3 K_COS
Wenn das TC1 Byte im ATR vorhanden ist, dann MUSS es den Wert 'FF' besitzen.**[<=]**

G2_N024.100.f - (N024.100)f K_COS
Das COS MUSS im TD1 Byte des ATR das Protokoll T=1 anzeigen.**[<=]**

G2_N024.100.g - (N024.100)g K_COS
Das COS DARF in den ATR KEIN TA2 Byte einstellen. Daraus folgt, dass das COS den "negotiable mode" unterstützt (siehe (N024.220)).**[<=]**

G2_N024.100.h - (N024.100)h K_COS
Das COS MUSS im TA3 Byte den Wert 254 = 'FE' enthalten. Dadurch wird gemäß [ISO/IEC 7816-3#11.4.2] eine "Information Field Size" IFSC = 254 = 'FE' angezeigt.**[<=]**

G2_N024.100.i - (N024.100).i K_COS
Das COS MUSS im ATR einen "class indicator" gemäß [ISO/IEC 7816-3#Tabelle 10] anzeigen. Der "class indicator" MUSS einen Wert aus der Menge {"A und B", "A, B und C"} = {3, 7} haben.**[<=]**

G2_N024.100.j - (N024.100)j K_COS
Wenn das COS im ATR in den Historical Bytes die "Third software function table" gemäß [ISO/IEC 7816-4#Table 119] enthält, dann MUSS die dort angezeigte Anzahl logischer Kanäle kleiner oder gleich der Anzahl maximal unterstützter logischer Kanäle sein.**[<=]**

(N024.200) Diese Anforderung ist absichtlich leer.

A_15727 - (N024.220) K_COS
Gemäß (N024.100)g zeigt der ATR stets den "negotiable mode" an. Deshalb MUSS das COS das "Protocol-and-Parameter-Selection"-Verfahren gemäß [ISO/IEC 7816-3#9] unterstützen. Im Einzelnen bedeutet das:**[<=]**

G2_N024.220.a - (N024.220)a K_COS
Wenn TA1 im ATR den Wert '18' besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {'12', '13', '18'} akzeptieren.**[<=]**

G2_N024.220.b - (N024.220)b K_COS
Wenn TA1 im ATR den Wert '95' besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {'92', '93', '94', '95'} akzeptieren.**[<=]**

G2_N024.220.c - (N024.220)c K_COS
Wenn TA1 im ATR den Wert '96 besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {'92', '93', '94', '95', '96'} akzeptieren.**[<=]**

G2_N024.220.d - (N024.220)d K_COS
Wenn TA1 im ATR den Wert '97 besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {'92', '93', '94', '95', '96', '97'} akzeptieren.**[<=]**

(N024.300) Diese Anforderung ist absichtlich leer.

G2_N024.320 - (N024.320) K_COS
Das COS MUSS bezüglich der Datenübertragung mittels des Blockprotokolls T=1 die Vorgaben aus [EMV®_Book-1#9.2.4] erfüllen und bezüglich der Fehlerbehandlung die Vorgaben aus [ISO/IEC 7816-3#11.6.3] erfüllen.**[<=]**

(N024.400) Diese Anforderung ist absichtlich leer.

(N024.500) Diese Anforderung ist absichtlich leer.

(N024.600) Diese Anforderung ist absichtlich leer.

11.2.2 Übertragungsprotokoll gemäß [ISO/IEC 7816-12]

Dieses Kapitel behandelt das kontaktbehaftete Übertragungsprotokoll gemäß [ISO/IEC 7816-12]. Dieses Übertragungsprotokolls wird von der Smartcard optional bereitgestellt.

(N024.700) Diese Anforderung ist absichtlich leer.

(N024.800) Diese Anforderung ist absichtlich leer.

A_15728 - (N024.810) K_IC, Option_USB_Schnittstelle
Das IC einer Smartcard MUSS das kontaktbehaftete Übertragungsprotokoll gemäß [ISO/IEC 7816-12] unterstützen, wobei gilt:**[<=]**

G2_N024.810.a - (N024.810)a K_IC, K_COS, Option_USB_Schnittstelle
Das IC und COS einer Smartcard MÜSSEN "Bulk Transfer" gemäß [ISO/IEC 7816-12#8.1] unterstützen.**[<=]**

A_15729 - (N024.810)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS neben "Bulk Transfer" weitere Übertragungsarten aus [ISO/IEC 7816-12] ~~KÖNNEN~~
1. unterstützt oder
2. ablehnt.**[<=]**

Für die Übertragungsgeschwindigkeit des hier betrachteten Protokolls gilt:

G2_N024.820.a - (N024.820)a K_IC, Option_USB_Schnittstelle
Das IC MUSS die Übertragungsgeschwindigkeit "Low Speed" unterstützen.**[<=]**

A_15730 - (N024.820)b K_TST

Für die funktionale Eignung MUSS es zulässig sein, dass das COS neben der Übertragungsgeschwindigkeit "Low Speed" weitere Übertragungsgeschwindigkeiten
1. unterstützt oder
2. ablehnt.**[<=]**

(N024.900) Diese Anforderung ist absichtlich leer.

(N025.000) Diese Anforderung ist absichtlich leer.

(N025.100) Diese Anforderung ist absichtlich leer.

11.2.3 Kontaktlose Datenübertragung gemäß ISO/IEC 14443

Dieses Kapitel behandelt die kontaktlose Datenübertragung gemäß ISO/IEC 14443. Die Unterstützung dieser Datenübertragungsart ist optional.

(N025.200) Diese Anforderung ist absichtlich leer.

(N025.300) Diese Anforderung ist absichtlich leer.

A_15731 - (N025.310) K_IC, Option_kontaktlose_Schnittstelle
Das IC einer Smartcard MUSS die kontaktlose Datenübertragung gemäß [ISO/IEC 14443-1], [ISO/IEC 14443-2], [ISO/IEC 14443-3] und [ISO/IEC 14443-4] unterstützen. Dabei gelten folgende Besonderheiten:**[<=]**

G2_N025.310.a - (N025.310)a K_IC, Option_kontaktlose_Schnittstelle
Das IC und das COS MÜSSEN mindestens einen Typ aus der Menge {"Typ A", "Typ B"} unterstützen.**[<=]**

G2_N025.310.b.1 - (N025.310)b.1 K_IC, Option_kontaktlose_Schnittstelle
Das IC und das COS MÜSSEN alle Datenraten aus der Menge {f_C / 128, f_C / 64, f_C / 32, f_C / 16} unterstützen.**[<=]**

A_15732 - (N025.310)b.2 K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS neben den in (N025.310)b.1 genannten Datenraten, weitere Datenraten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N025.310.c.1 - (N025.310)c.1 K_IC, K_COS, K_Antenne, Option_kontaktlose_Schnittstelle
Die gesamte Karte bestehend aus IC, COS und Antenne im Kartenkörper MUSS alle Datenraten aus der Menge {f_C / 128} unterstützen.**[<=]**

G2_N025.310.c.2 - (N025.310)c.2 K_IC, K_COS, K_Antenne, Option_kontaktlose_Schnittstelle
Die gesamte Karte bestehend aus IC, COS und Antenne im Kartenkörper SOLL alle Datenraten aus der Menge {f_C / 64, f_C / 32, f_C / 16} unterstützen.**[<=]**

A_15733 - (N025.310)c.3 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass eine Karte neben den in (N025.310)c.1 und (N025.310)c.2 genannten Datenraten weitere Datenraten
i. unterstützt oder
ii. ablehnt.**[<=]**

A_15734 - (N025.320) K_IC, K_COS, Option_kontaktlose_Schnittstelle

Das IC einer Smartcard und das COS MÜSSEN bei der kontaktlosen Datenübertragung spätestens mit Beginn der Nutzungsphase (siehe Kapitel 4) dynamisch generierte, zufällig gewählte Kennnummern verwenden, siehe dazu
a. [ISO/IEC 14443-3#6.5.4] für "Typ A" Smartcard und
b. [ISO/IEC 14443-3#7.9.2] für "Typ B" Smartcard**[<=]**

(N025.400) Diese Anforderung ist absichtlich leer.

(N025.500) Diese Anforderung ist absichtlich leer.

(N025.510) Diese Anforderung ist absichtlich leer.

11.3 OSI-Referenzmodell (informativ)

In Anlehnung an das OSI-Referenzmodell werden hier verschiedene Layer definiert, die an der Bearbeitung einer Nachricht beteiligt sind, welche von einem externen Kommunikationspartner gesendet wurde. Es sei ausdrücklich darauf hingewiesen, dass die Inhalte dieses Abschnittes keine Implementierungsdetails festlegen. Das bedeutet, es ist zulässig, den internen Aufbau eines Smartcard-Betriebssystems oder dessen Kommunikationsaufbau anders zu gestalten. Die normativen Teile dieses Dokumentes lassen sich aber leichter beschreiben und verstehen, wenn man das Folgende zugrunde legt.

In CosA_e09 wird exemplarisch gezeigt, wie eine Kommando-APDU CmdApdu1 vom externen Kommunikationspartner zunächst entsprechend den Konventionen des Übertragungsprotokolls (siehe 11.2) im physikalischen Layer und im Data Link Layer in einen oder möglicherweise mehrere TPDU zerlegt und im I/O der Smartcard wieder zu einer CmdApdu1 gemäß 11.5 zusammengesetzt wird.

Der nächste Layer verwaltet logische Kanäle (siehe [ISO/IEC 7816-4#5.4.2]) und leitet das empfangene Kommando entsprechend der Kanalnummer im CLA Byte weiter an den entsprechenden logischen Kanal, hier Channel_x.

Im nächsten Layer "SecMes" wird die optional per Secure Messaging gesicherte Kommando-APDU ausgepackt und an den Kommandointerpreter weitergeleitet.

Der Kommandointerpreter verarbeitet die Kommando-APDU und erstellt eine entsprechende Antwort-APDU, welche optional per Secure Messaging gesichert wird. Die eventuell gesicherte Antwort wird im I/O in eine oder mehrere TPDU zerlegt, welche dann über die physikalische Schnittstelle an den externen Kommunikationspartner übermittelt werden.

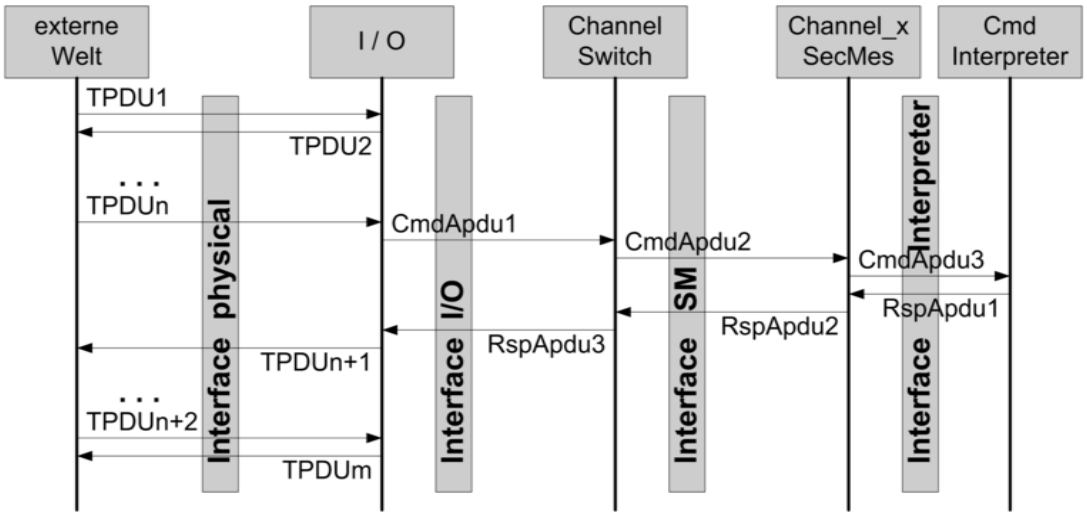


Abbildung 1, CosA_e09: Message Sequence Chart für die Kommandobearbeitung

11.4 Kommandobearbeitung

Dieses Kapitel beschreibt die Bearbeitung eines Kommandos, welches von einem externen Kommunikationspartner an das COS gesendet wurde. Es werden die Begriffe aus 11.3 verwendet. Es wird nochmals darauf verwiesen, dass die Inhalte von 11.3 nicht normativ sind. Insofern ist die genaue Zuordnung der normativen Aussagen dieses Kapitels zu Teilen eines COS nicht festgelegt. Wichtig ist aber, dass das Verhalten des COS auf Ebene der physikalischen Schnittstelle den normativen Vorgaben entspricht.

Bei der Bearbeitung einer Kommando-APDU und der Generierung einer Antwort-APDU werden folgende Schritte durchlaufen:

G2_N025.600 - (N025.600) K_externeWelt {K_Karte}
Ein externer Kommunikationspartner sendet über eine physikalische Schnittstelle eine Kommando-APDU CmdApdu1, welche den Vorgaben aus 11.5 entsprechen MUSS. Dazu wird CmdApdu1 auf der physikalischen Schnittstelle entsprechend dem Übertragungsprotokoll (siehe 11.2) als (eine oder mehrere) TPDU transportiert.**[<=]**

G2_N025.650 - (N025.650) K_COS
Das I/O der Smartcard MUSS die TPDU zusammensetzen und gewinnt dabei CmdApdu1 zurück.**[<=]**

G2_N025.700 - (N025.700) K_COS
Das COS MUSS die Fehlerbehandlung auf der physikalischen Schnittstelle gemäß [EMV®_Book-1] umsetzen.**[<=]**

G2_N025.800 - (N025.800) K_COS
Das COS MUSS anschließend die Kanalnummer aus dem CLA Byte von CmdApdu1 extrahieren.**[<=]**

G2_N025.900.a - (N025.900)a K_COS

Wenn der Kanal mit der Kanalnummer aus dem CLA Byte nicht geöffnet ist, dann MUSS die Bearbeitung des Kommandos terminieren.**[<=]**

G2_N025.900.b - (N025.900)b K_COS
Wenn der Kanal mit der Kanalnummer aus dem CLA Byte nicht geöffnet ist, dann MUSS die RspApdu3 lediglich aus dem Trailer ChannelClosed = '68 81' bestehen. Dies Response APDU enthält in diesem Fall keine Antwortdaten.**[<=]**

G2_N026.000 - (N026.000) K_COS
Entsprechend der Kanalnummer MUSS CmdApdu1 unverändert als CmdApdu2 an den entsprechenden logischen Kanal weitergeleitet werden.**[<=]**

(N026.100) Dieser Punkt ist absichtlich leer.

G2_N026.200 - (N026.200) K_COS
Das COS MUSS innerhalb des logischen Kanals CmdApdu2 mit dem entsprechenden Kanalkontext *channelContext* an den Secure Messaging Layer ("SecMes" in CosA_e09) weiterleiten und dort gemäß13.1.2 bearbeiten. Das Ergebnis ist RspApdu2.**[<=]**

G2_N026.300 - (N026.300) K_COS
Das COS MUSS RspApdu2 MUSS unverändert als RspApdu3 an das I/O der Smartcard schicken.**[<=]**

G2_N026.400 - (N026.400) K_COS
Das I/O der Smartcard MUSS RspApdu3 gemäß dem verwendeten Übertragungsprotokoll in eine oder mehrere TPDU umwandeln und über die physikalische Schnittstelle senden.**[<=]**

11.5 Kommando-APDU

11.5.1 Class Byte

Das Class Byte (CLA Byte) enthält die Kommandoklasse. Zusammen mit dem Instruction Byte (siehe 11.5.2) legt es eindeutig das auszuführende Kommando fest.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

G2_N026.500 - (N026.500) K_externeWelt {K_Karte}
Gemäß [ISO/IEC 7816-3] MUSS das CLA Byte in einem Oktett codiert werden.**[<=]**

G2_N026.510 - (N026.510) K_COS
Die Bits im CLA Byte MÜSSEN die Bedeutung gemäß CosT_760 oder CosT_67a haben.

Tabelle 70: CosT_760: Codierung des CLA-Bytes für Kanalnummern kleiner vier

b8	b7	b6	b5	b4	b3	b2	b1	Beschreibung
x	0	0	-	-	-	-	-	**Unterscheidung der Kommandoklasse**
0	0	0	-	-	-	-	-	Kommando gemäß ISO/IEC 7816
1	0	0	-	-	-	-	-	Kommandoerweiterung
-	0	0	x	-	-	-	-	**Command Chaining Indikator**
-	0	0	0	-	-	-	-	Einziges Kommando oder letztes Kommando einer Kette
-	0	0	1	-	-	-	-	Nicht letztes Kommando einer Kette
-	0	0	-	x	x	-	-	**Secure Messaging Indikator**
-	0	0	-	1	1	-	-	Secure Messaging gemäß 13.1.2
-	0	0	-	0	0	-	-	Kein Secure Messaging
-	0	0	-	-	-	x	x	**Kanalnummer gemäß [ISO/IEC 7816-4#Tab. 2]**

Tabelle 71: CosT_67a: Codierung des CLA-Bytes für Kanalnummern größer gleich vier

b8	b7	b6	b5	b4	b3	b2	b1	Beschreibung
x	1	-	-	-	-	-	-	**Unterscheidung der Kommandoklasse**
0	1	-	-	-	-	-	-	Kommando gemäß ISO/IEC 7816
1	1	-	-	-	-	-	-	Kommandoerweiterung
-	1	x	-	-	-	-	-	**Secure Messaging Indikator**
-	1	1	-	-	-	-	-	Secure Messaging gemäß 13.1.2
-	1	0	-	-	-	-	-	Kein Secure Messaging
-	1	-	x	-	-	-	-	**Command Chaining Indikator**
-	1	-	0	-	-	-	-	Einziges Kommando oder letztes Kommando einer Kette
-	1	-	1	-	-	-	-	Nicht letztes Kommando einer Kette
-	1	-	-	x	x	x	x	**Kanalnummer gemäß [ISO/IEC 7816-4#Tab. 3]**

[<=]

11.5.2 Instruction Byte

Das Instruction Byte (INS Byte) enthält die Codierung für den auszuführenden Befehl. Zusammen mit dem Class Byte (siehe 11.5.1) legt es eindeutig das auszuführende Kommando fest.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

A_15735 - (N026.600) K_externeWelt {K_Karte}
Gemäß [ISO/IEC 7816-3] MUSS das INS Byte in einem Oktett codiert werden.**[<=]**

11.5.3 Parameter P1

Der Parameter P1 enthält in der Regel eine Variable, die bei der Kommandoausführung benötigt wird. Die Bedeutung dieses Parameters hängt gewöhnlich von der Kombination aus CLA, INS und P2 ab.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

A_15736 - (N026.700) K_externeWelt {K_Karte}
Gemäß [ISO/IEC 7816-3] MUSS der Parameter P1 in einem Oktett codiert werden.**[<=]**

11.5.4 Parameter P2

Der Parameter P2 enthält in der Regel eine Variable, die bei der Kommandoausführung benötigt wird. Die Bedeutung dieses Parameters hängt von der Kombination aus CLA, INS und P1 ab.

Welche Kombinationen aus CLA, INS, P1 und P2 zu unterstützen sind, wird in Kapitel 14 festgelegt.

A_15737 - (N026.800) K_externeWelt {K_Karte}
Gemäß [ISO/IEC 7816-3] MUSS der Parameter P2 in einem Oktett codiert werden.**[<=]**

11.5.5 Datenfeld

Das Datenfeld einer Kommando-APDU ist optional. Es ist möglich, dass es fehlt. Das Datenfeld ist ein Oktettstring der Länge Nc. Gemäß [ISO/IEC 7816-3] ist der Definitionsbereich von Nc gleich {1, …, 65535}.

G2_N026.900.a - (N026.900)a K_COS
Das COS MUSS für Nc an der Schnittstelle "Interpreter" (siehe CosA_e09) alle Werte unterstützen, die sich aus (N029.892) ergeben.**[<=]**

A_15738 - (N026.900)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS über den durch (N029.892) definierten Bereich hinaus weitere Werte für Nc
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N026.900.c - (N026.900)c K_COS
Ein COS DARF eine APDU NICHT wegen eines Längenfehlers an der Schnittstelle "Interpreter" (siehe CosA_e09) abweisen, wenn
1. die Kommando-APDU den Anforderungen aus 11.7 genügt und
2. das Datenfeld die Längenbeschränkung aus (N026.900)a oder die gesamte Kommando-APDU die Längenbeschränkung aus EF.ATR erfüllt.**[<=]**

A_15739 - (N026.910) K_externeWelt {K_Karte}
Der Sender einer Kommando-APDU ~~SOLL~~MUSS die Längenbeschränkung durch passende Wahl von Nc einhalten.**[<=]**

Hinweis CosH_576: Die obere Grenze in (N026.900)a gilt für das COS einer Smartcard. Für das IFD (Kartenleser) wird wegen möglicher, zukünftiger Erweiterungen, Entwicklungen und wegen Performanzgewinns empfohlen, größere Werte zu unterstützen.

Hinweis CosH_ce0: Anders als in der Generation 1 ist die obere Grenze in (N026.900)a nicht mehr durch ein bestimmtes Kommando motiviert, welches besonders viele Daten überträgt und sich zudem schlecht auf mehrere Kommandos aufteilen lässt. Die in die (N026.900)a genannte Grenze ergibt sich aus Performanzüberlegungen verbunden mit dem aktuellen Stand der Technik.

11.5.6 LeFeld

Das LeFeld einer Kommando-APDU ist optional. Es ist möglich, dass es fehlt. Das LeFeld enthält eine Zahl Ne, die angibt, wie viele Oktette im Datenfeld der Antwort-APDU erwartet werden. In diesem Dokument gelten folgende Definitionen:

Tabelle 72: CosT_93c: Definition von WildCard-Werten für das LeFeld

WildCardShort	Dieser Wert wird verwendet, um anzuzeigen, dass innerhalb der Obergrenze von 256 Oktetten alle verfügbaren Oktette in das Datenfeld der Antwortnachricht einzustellen sind.
WildCardExtended	Dieser Wert wird verwendet, um anzuzeigen, dass innerhalb der Obergrenze von 65.536 Oktetten alle verfügbaren Oktette in das Datenfeld der Antwortnachricht einzustellen sind.

Gemäß [ISO/IEC 7816-4] ist der Definitionsbereich für Ne: {1, 2, …, 65.534, 65.535, WildCardShort, WildCardExtended}

G2_N027.000.a - (N027.000)a K_COS
Das COS MUSS an der Schnittstelle "Interpreter" (siehe CosA_e09) für Ne alle Werte aus der Menge {1, 2, …, 65.534, 65.535, WildCardShort, WildCardExtended} unterstützen.**[<=]**

G2_N027.000.b - (N027.000)b K_COS
Ein COS DARF eine APDU NICHT wegen eines Längenfehlers an der Schnittstelle "Interpreter" (siehe CosA_e09) abweisen, wenn die Längenbeschränkung aus (N027.000)a oder die gesamte Antwort-APDU die Längenbeschränkung aus EF.ATR erfüllt.**[<=]**

A_15740 - (N027.010) K_externeWelt {K_Karte}
Der Sender einer Kommando-APDU ~~SOLL~~MUSS die Längenbeschränkung durch passende Wahl von Ne einhalten.**[<=]**

Hinweis CosH_317: Die obere Grenze in (N027.000)a gilt für das COS einer Smartcard und (N027.010) gilt für die "externe Welt". Für das IFD (Kartenleser) wird wegen Performanzgewinns empfohlen, ebenfalls die gemäß [ISO/IEC 7816-3] maximal mögliche Länge einer Antwortnachricht zu unterstützen.

Hinweis CosH_b79: Anders als in der Generation 1 ist die obere Grenze in (N027.000)a nicht mehr durch ein bestimmtes Kommando motiviert, welches besonders viele Daten überträgt und sich zudem schlecht auf mehrere Kommandos aufteilen lässt. Im Rahmen dieses Dokumentes ist es das READ BINARY *Kommando, welches die größtmögliche Antwortdatenmenge überträgt.*

11.6 Antwort-APDU

Mit den Definitionen aus den 11.6.1 und 11.6.2 ergibt sich für eine Antwort-APDU:

G2_N027.100 - (N027.100) K_COS
Eine Antwort-APDU MUSS ein Oktettstring gemäß '*rspData* || Trailer' sein, wobei *rspData* möglicherweise leer ist.**[<=]**

11.6.1 Datenfeld

Das Datenfeld *rspData* einer Antwort-APDU ist optional in dem Sinne, dass es möglicherweise leer ist~~. Es ist möglich, dass es fehlt.~~ Das Datenfeld *rspData* ist ein Oktettstring der Länge Nr. Gemäß [ISO/IEC 7816-3] ist der Definitionsbereich von Nr {0, 1, …, 65536}.

A_15787 - (N027.200) K_COS
An der Schnittstelle "Interpreter" (siehe CosA_e09) gilt für Nr: Das COS MUSS dafür sorgen, dass Nr kleiner gleich Ne der zugehörigen Kommando-APDU sein.**[<=]**

G2_N027.200.a - (N027.200)a K_COS
Wenn eine Kommando-APDU kein LeFeld besitzt, dann MUSS das COS dafür sorgen, dass Nr = 0 ist.**[<=]**

G2_N027.200.b.1 - (N027.200)b.1 K_COS
Wenn eine Kommando-APDU ein LeFeld = '00' = WidlCardShort besitzt, dann MUSS das COS dafür sorgen, dass Nr kleiner gleich '100' = 256 ist.**[<=]**

G2_N027.200.b.2 - (N027.200)b.2 K_COS
Wenn eine Kommando-APDU ein LeFeld = '0000' = WildCardExtended besitzt, dann MUSS das COS dafür sorgen, dass Nr kleiner gleich '10000' = 65536 ist.**[<=]**

G2_N027.200.b.3 - (N027.200)b.3 K_COS
Wenn eine Kommando-APDU ein LeFeld besitzt und dieses nicht Element der Menge {'00', '0000'} ist, dann MUSS das COS dafür sorgen, dass Nr kleiner gleich OS2I(LeFeld) ist.**[<=]**

11.6.2 Trailer

Der Trailer zeigt den Status nach Bearbeitung einer Kommandonachricht an. CosT_7aa zeigt die in diesem Dokument definierten Trailer. Der Trailer enthält Statusangaben über die Bearbeitung des Kommandos. Dazu zählen auch Fehlerindikationen.

A_15788 - (N027.210) K_COS
Das COS MUSS den Trailer (siehe [ISO/IEC 7816-4#Tabelle 1]) in zwei Oktetten codieren.**[<=]**

11.7 Zulässige Kommando-Antwort-Paare

Wie in 11.5.5 und 11.5.6 dargestellt, sind zwei Bestandteile einer Kommando-APDU optional. Daraus ergeben sich vier Kombinationsmöglichkeiten, die im Folgenden beschrieben werden.

11.7.1 Case 1 Kommando-Antwort-Paar

Im Fall einer Case 1 Kommando-APDU fehlen Datenfeld und LeFeld. Die Kommando-APDU enthält folgende Angaben:

Tabelle 73: CosT_2be: Case 1 Kommando-APDU

	Inhalt	Beschreibung
CLA	'XX'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'XX'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	Erster Parameter
P2	'XX'	Zweiter Parameter

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 1 Kommando-APDU:

A_15789 - (N027.300) K_externeWelt {K_Karte}
Eine Case 1 Kommando-APDU MUSS aus vier Oktetten bestehen.**[<=]**

A_15790 - (N027.400) K_externeWelt {K_Karte}
Die vier Oktette einer Case 1 Kommando-APDU MÜSSEN wie folgt konkateniert werden:
CLA || INS || P1 || P2.**[<=]**

Die zu einer Case 1 Kommando-APDU gehörende Antwort-APDU besteht nur aus dem Trailer.

Tabelle 74: CosT_d12: Case 1 Antwort-APDU

Inhalt	Beschreibung
Trailer	Statusbytes SW1 und SW2

11.7.2 Case 2 Kommando-Antwort-Paar

Im Fall einer Case 2 Kommando-APDU fehlt das Datenfeld. Das LeFeld ist vorhanden. In Abhängigkeit vom Wert Ne, der im LeFeld transportiert wird, werden die Fälle "short" und "extended" unterschieden.

11.7.2.1 Case 2 Short Kommando

In diesem Fall enthält das Datenfeld der Antwortnachricht nie mehr als 256 Oktette. Die Kommando-APDU enthält im Fall einer Case 2 Short folgende Angaben:

Tabelle 75: CosT_55c: Case 2 Short Kommando-APDU

	Inhalt	Beschreibung
CLA	'XX'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'XX'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	Erster Parameter
P2	'XX'	Zweiter Parameter
Le	'XX'	Ne aus der Menge {1, …, 255, WildCardShort}

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 2 Short Kommando-APDU:

A_15791 - (N027.500) K_externeWelt {K_Karte}
Eine Case 2 Short Kommando-APDU MUSS aus fünf Oktetten bestehen.**[<=]**

A_15792 - (N027.600) K_externeWelt {K_Karte}
Der Wert von Ne wird wie folgt in einem Oktett codiert, welches als LeFeld bezeichnet wird:
a. Wenn Ne im Intervall [1, 255] liegt, dann MUSS diese ganze Zahl in einem Oktett codiert werden: LeFeld = I2OS(Ne, 1).
b. Wenn Ne den Wert WildCardShort besitzt, dann MUSS das LeFeld gleich '00' sein.**[<=]**

A_15793 - (N027.700) K_externeWelt {K_Karte}
Die fünf Oktette einer Case 2 Short Kommando-APDU MÜSSEN wie folgt konkateniert werden:
CLA || INS || P1 || P2 || LeFeld.**[<=]**

11.7.2.2 Case 2 Extended Kommando

In diesem Fall enthält das Datenfeld der Antwortnachricht möglicherweise mehr als 256 Oktette. Die Kommando-APDU enthält im Fall einer Case 2 Extended folgende Angaben:

Tabelle 76: CosT_12e: Case 2 Extended Kommando-APDU

	Inhalt	Beschreibung
CLA	'XX'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'XX'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	Erster Parameter
P2	'XX'	Zweiter Parameter
Le	'XXXX'	Ne aus der Menge {256, …, 65535, WildCardExtended}

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 2 Extended Kommando-APDU:

A_15794 - (N027.800) K_externeWelt {K_Karte}
Eine Case 2 Extended Kommando-APDU MUSS aus sieben Oktetten bestehen.**[<=]**

A_15795 - (N027.900) K_externeWelt {K_Karte}
Der Wert von Ne wird wie folgt in zwei Oktetten codiert, welche das LeFeld bilden:
a. Wenn Ne im Intervall [256, 65535] liegt, dann MUSS diese ganze Zahl in zwei Oktette codiert werden: LeFeld = I2OS(Ne, 2).
b. Wenn Ne den Wert WildCardExtended besitzt, dann MUSS das LeFeld gleich '0000' sein.**[<=]**

A_15796 - (N028.000) K_externeWelt {K_Karte}
Die sieben Oktette einer Case 2 Extended Kommando-APDU MÜSSEN wie folgt konkateniert werden:
CLA || INS || P1 || P2 || '00' || LeFeld.**[<=]**

Hinweis CosH_d25: Das Oktett '00' nach dem Parameter P2 lässt sich als Indikator für "extended length" auffassen.

11.7.2.3 Case 2 Response

Die zu einer Case 2 Kommando-APDU gehörende Antwort-APDU besteht aus dem optionalen Datenfeld und aus dem Trailer.

Tabelle 77: CosT_e50: Case 2 Antwort-APDU

Inhalt	Beschreibung
Daten	Optionaler Bestandteil der Antwort-APDU. Wenn vorhanden, dann gelten die Bestimmungen aus 11.6.1.
Trailer	Statusbytes SW1 und SW2

11.7.3 Case 3 Kommando-Antwort-Paare

Im Fall einer Case 3 Kommando-APDU fehlt das LeFeld. Das Datenfeld ist vorhanden. In Abhängigkeit von der Anzahl der Oktette im Datenfeld werden die Fälle "short" und "extended" unterschieden.

11.7.3.1 Case 3 Short Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht nie mehr als 255 Oktette. Die Kommando-APDU enthält im Fall einer Case 3 Short folgende Angaben:

Tabelle 78: CosT_fbc: Case 3 Short Kommando-APDU

	Inhalt	Beschreibung
CLA	'XX'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'XX'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	Erster Parameter
P2	'XX'	Zweiter Parameter
Data	'XX…XX'	Datenfeld mit beliebigen Oktetten, Anzahl Oktette aus dem Intervall [1, 255]

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 3 Short Kommando-APDU:

A_15797 - (N028.100) K_externeWelt {K_Karte}
Eine Case 3 Short Kommando-APDU MUSS aus fünf Oktetten plus den Oktetten aus dem Datenfeld bestehen.**[<=]**

A_15798 - (N028.200) K_externeWelt {K_Karte}
Die Anzahl Nc der Oktette des Datenfeldes MUSS in einem Oktett codiert werden, welches als LcFeld bezeichnet wird:
LcFeld = I2OS(Nc, 1).**[<=]**

A_15799 - (N028.300) K_externeWelt {K_Karte}
Die fünf plus Nc Oktette einer Case 3 Short Kommando-APDU MÜSSEN wie folgt konkateniert werden:
CLA || INS || P1 || P2 || LcFeld || Datenfeld.**[<=]**

11.7.3.2 Case 3 Extended Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht mehr als 255 Oktette. Die Kommando-APDU enthält im Fall einer Case 3 Extended folgende Angaben:

Tabelle 79: CosT_740: Case 3 Extended Kommando-APDU

	Inhalt	Beschreibung
CLA	'XX'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'XX'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	Erster Parameter
P2	'XX'	Zweiter Parameter
Data	'XX…XX'	Datenfeld mit beliebigem Inhalt, Anzahl Oktette aus dem Intervall [256, 65535]

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 3 Extended Kommando-APDU:

A_15800 - (N028.400) K_externeWelt {K_Karte}
Eine Case 3 Extended Kommando-APDU MUSS aus sieben Oktetten plus den Oktetten aus dem Datenfeld bestehen.**[<=]**

A_15801 - (N028.500) K_externeWelt {K_Karte}
Die Anzahl Nc der Oktette des Datenfeldes MUSS in zwei Oktette codiert werden, welche als LcFeld bezeichnet werden:
LcFeld = I2OS(Nc, 2).**[<=]**

A_15802 - (N028.600) K_externeWelt {K_Karte}
Die sieben plus Nc Oktette einer Case 3 Extended Kommando-APDU MÜSSEN wie folgt konkateniert werden:
CLA || INS || P1 || P2 || '00' || LcFeld || Datenfeld.**[<=]**

Hinweis CosH_7c2: Das Oktett '00' nach dem Parameter P2 lässt sich als Indikator für "extended length" auffassen.

11.7.3.3 Case 3 Response

Die zu einer Case 3 Kommando-APDU gehörende Antwort-APDU besteht nur aus dem Trailer.

Tabelle 80: CosT_42d: Case 3 Antwort-APDU

Inhalt	Beschreibung
Trailer	Statusbytes SW1 und SW2

11.7.4 Case 4 Kommando-Antwort-Paare

Eine Case 4 Kommando-APDU enthält alle optionalen Bestandteile. In Abhängigkeit von der Anzahl der Oktette im Datenfeld der Kommandonachricht und in Abhängigkeit vom Wert Ne, der im LeFeld transportiert wird, werden die Fälle "short" und "extended" unterschieden.

11.7.4.1 Case 4 Short Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht nie mehr als 255 Oktette und das Datenfeld der Antwortnachricht nie mehr als 256 Oktette. Die Kommando-APDU enthält im Fall einer Case 4 Short folgende Angaben:

Tabelle 81: CosT_704: Case 4 Short Kommando-APDU

	Inhalt	Beschreibung
CLA	'XX'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'XX'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	Erster Parameter
P2	'XX'	Zweiter Parameter
Data	'XX…XX'	Datenfeld mit beliebigen Oktetten, Anzahl Oktette aus dem Intervall [1, 255]
Le	'XX'	Ne aus der Menge {1, …, 255, WildCardShort}
Notation		CmdApdu = Case4S(CLA, INS, P1, P2, Data, Ne)

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 4 Short Kommando-APDU:

A_15803 - (N028.700) K_externeWelt {K_Karte}
Eine Case 4 Short Kommando-APDU MUSS aus sechs Oktetten plus den Oktetten aus dem Datenfeld bestehen.**[<=]**

A_15804 - (N028.800) K_externeWelt {K_Karte}
Die Anzahl Nc der Oktette des Datenfeldes MUSS in einem Oktett codiert werden, welches als LcFeld bezeichnet wird:
LcFeld = I2OS(Nc, 1).**[<=]**

A_15805 - (N028.900) K_externeWelt {K_Karte}
Der Wert von Ne wird wie folgt in einem Oktett codiert, welches das LeFeld bildet:
a. Wenn Ne im Intervall [1, 255] liegt, dann MUSS diese ganze Zahl in einem Oktett codiert werden: LeFeld = I2OS(Ne, 1).
b. Wenn Ne den Wert WildCardShort besitzt, dann MUSS das LeFeld gleich '00' sein.**[<=]**

A_15806 - (N029.000) K_externeWelt {K_Karte}
Die sechs plus Nc Oktette einer Case 4 Short Kommando-APDU MÜSSEN wie folgt konkateniert werden:
CLA || INS || P1 || P2 || LcFeld || Datenfeld || LeFeld.**[<=]**

11.7.4.2 Case 4 Extended Kommando

In diesem Fall enthält das Datenfeld der Kommandonachricht mehr als 255 Oktette oder es wird erwartet, dass das Datenfeld der Antwortnachricht mehr als 256 Oktette enthält. Die Kommando-APDU enthält im Fall einer Case 4 Extended folgende Angaben:

Tabelle 82: CosT_447: Case 4 Extended Kommando-APDU

	Inhalt	Beschreibung
CLA	'XX'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'XX'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	Erster Parameter
P2	'XX'	Zweiter Parameter
Data	'XX…XX'	Datenfeld mit beliebigen Oktetten, Anzahl Oktette aus dem Intervall [1, 65535]
Le	'XXXX'	Ne aus der Menge {1, …, 65535, WildCardExtended}
Notation		CmdApdu = Case4E(CLA, INS, P1, P2, Data, Ne)

Gemäß den Regeln aus 11.5 werden CLA, INS, P1 und P2 in jeweils einem Oktett codiert. Damit gilt für die gesamte Case 4 Extended Kommando-APDU:

A_15807 - (N029.100) K_externeWelt {K_Karte}
Eine Case 4 Extended Kommando-APDU MUSS aus neun Oktetten plus den Oktetten aus dem Datenfeld bestehen.**[<=]**

A_15808 - (N029.200) K_externeWelt {K_Karte}
Die Anzahl Nc der Oktette des Datenfeldes MUSS in zwei Oktetten codiert werden, welche als LcFeld bezeichnet werden:
LcFeld = I2OS(Nc, 2).**[<=]**

A_15809 - (N029.300) K_externeWelt {K_Karte}
Der Wert von Ne wird wie folgt in zwei Oktette codiert, welche das LeFeld bilden:
a. Wenn Ne im Intervall [1, 65535] liegt, dann MUSS diese ganze Zahl in zwei Oktette codiert werden: LeFeld = I2OS(Ne, 2).
b. Wenn Ne den Wert WildCardExtended besitzt, dann MUSS das LeFeld gleich '0000' sein.**[<=]**

A_15810 - (N029.400) K_externeWelt {K_Karte}
Die neun plus Nc Oktette einer Case 4 Extended Kommando-APDU MÜSSEN wie folgt konkateniert werden:
CLA || INS || P1 || P2 || '00' || LcFeld || Datenfeld || LeFeld.**[<=]**

Hinweis CosH_67b: Das Oktett '00' nach dem Parameter P2 lässt sich als Indikator für "extended length" auffassen.

11.7.4.3 Case 4 Response

Die zu einer Case 4 Kommando-APDU gehörende Antwort-APDU besteht aus dem optionalen Datenfeld und aus dem Trailer.

Tabelle 83: CosT_d0a: Case 4 Antwort-APDU

Inhalt	Beschreibung
Daten	Optionaler Bestandteil der Antwort-APDU. Wenn vorhanden, dann gelten die Bestimmungen aus 11.6.1.
Trailer	Statusbytes SW1 und SW2

(N029.500), (N029.600), (N029.610), (N029.700), (N029.800) Diese Punkte sind absichtlich leer.

11.8 Command Chaining

Gemäß [ISO/IEC 7816-4#5.3.3] ist "Command Chaining" die Möglichkeit, auszudrücken, dass konsekutive Kommando-Antwort-Paare verkettet sind, also zusammen gehören. In diesem Sinne wird der Mechanismus genutzt, um die Teilschritte eines mehrschrittigen Ablaufes als zusammengehörig zu kennzeichnen. Während in [ISO/IEC 7816-4#5.3.3] einige Details offen gelassen werden, gilt für dieses Dokument folgendes Verhalten:

Command Chaining ist eine konsekutive Folge von Kommando-Antwort-Paaren mit folgenden Eigenschaften:

G2_N029.870.a - (N029.870)a K_externeWelt {K_Karte}
Mit Ausnahme des Bits b5 im CLA-Byte MÜSSEN alle Kommando-APDU einer Chaining-Kette identische Werte für CLA, INS, P1 und P2 haben.**[<=]**

G2_N029.870.b - (N029.870)b K_externeWelt {K_Karte}
Das Bit b5 im CLA-Byte der letzten Kommando-APDU einer Command-Chaining-Sequenz MUSS den Wert 0 und die Bits b5 aller anderen CLA-Bytes einer Command-Chaining-Sequenz MÜSSEN den Wert 1 haben.**[<=]**

G2_N029.870.c - (N029.870)c K_externeWelt {K_Karte}
Die Aussagen in (N029.870)a und (N029.870)b MÜSSEN für Kommando-APDU an der Schnittstelle "Interface I/O" aus CosA_e09 gelten.**[<=]**

A_15811 - (N029.874) K_externeWelt {K_Karte}
Die konsekutive Folge einer Command-Chaining-Sequenz ~~SOLL~~DARF an der Schnittstelle "Interface I/O" aus CosA_e09 NICHT
a. durch Kommandos unterbrochen werden, welche nicht zu dieser Sequenz gehören oder
b. durch eine Deaktivierung unterbrochen werden.**[<=]**

A_15812 - (N029.876) K_externeWelt {K_Karte}
Die konsekutive Folge einer Command-Chaining-Sequenz DARF an der Schnittstelle "Interface I/O" aus CosA_e09 NICHT fortgesetzt werden, falls ein Kommando der Sequenz mit einem Fehler terminiert.**[<=]**

G2_N029.876.a - (N029.876)a K_externeWelt {K_Karte}
Wenn LOAD APPLICATION mit einem Trailer aus CosT_df0 antwortet, dann DARF die Command-Chaining-Sequenz an der Schnittstelle "Interface I/O" aus CosA_e09 NICHT fortgesetzt werden.**[<=]**

G2_N029.876.b - (N029.876)b K_externeWelt {K_Karte}
Wenn GENERAL AUTHENTICATE mit einem Trailer aus CosT_b4e antwortet, dann DARF die Command-Chaining-Sequenz an der Schnittstelle "Interface I/O" aus CosA_e09 NICHT fortgesetzt werden.**[<=]**

G2_N029.878.a - (N029.878)a K_COS
Wenn die Anforderung (N029.874)a von der externen Entität nicht eingehalten wird, mithin also die Command-Chaining-Sequenz durch ein Kommando unterbrochen wird, dann MUSS das COS das unterbrechende Kommando akzeptieren.**[<=]**

A_15813 - (N029.878)b K_TST

~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS das Fortsetzen einer unterbrochenen Comman-Chaining-Sequenz akzeptiert.**[<=]**

G2_N029.878.c - (N029.878)c K_COS
Wenn die Anforderung (N029.874)a von der externen Entität nicht eingehalten wird, mithin also die Command-Chaining-Sequenz durch ein Kommando unterbrochen wird, dann SOLL das COS ein Fortsetzen der unterbrochenen Sequenz ablehnen.**[<=]**

G2_N029.880.a - (N029.880)a K_COS
Wenn die Anforderung (N029.876) von der externen Entität nicht eingehalten wird, mithin also die Command-Chaining-Sequenz trotz Fehlers im vorherigen Kommando fortgesetzt wird, dann SOLL das COS ein Fortsetzen der Sequenz ablehnen.**[<=]**

A_15814 - (N029.880)b K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS das Fortsetzen einer unterbrochenen Command-Chaining-Sequenz trotz Fehlers im vorherigen Kommando akzeptiert.**[<=]**

Hinweis CosH_8f7: Es ist gewollt, dass gemäß (N029.870)a alle Kommandos einer Kette dieselbe Kanalnummer aufweisen und die Secure Messaging Bits im CLA-Byte identisch sind. In Verbindung mit (N029.874) folgt daraus, dass Command Chaining nicht durch Kommandos auf anderen logischen Kanälen unterbrechbar ist.

11.9 Längenbeschränkung von APDU

Gemäß [ISO/IEC 7816-4] ist die Länge einer APDU auf rund 64 kByte beschränkt. Zumindest für Kommando-APDU entspricht es dem Stand der Technik, dass diese Grenze wegen der RAM-Speichergrößen heute verfügbarer Halbleiter bei weitem nicht ausgeschöpft wird. Typischerweise zeigt eine Smartcard im EF.ATR an, welche APDU-Längenbeschränkung für diese Smartcard gilt. Dabei werden Werte getrennt nach Kommando- und Antwortnachricht einerseits und andererseits getrennt nach ungeschützter Übertragung und geschützter Übertragung angegeben.

Im folgenden werden Intervalle für die Länge einer Nachricht angegeben. Dabei bedeutet "ungeschützte Kommandonachricht", dass im CLA-Byte kein Secure Messaging angezeigt wird. In diesem Fall wird die korrespondierende Antwortnachricht ebenfalls ungeschützt übertragen. Im Falle einer "geschützten Kommandonachricht" wird im CLA-Byte Secure Messaging angezeigt und die korrespondierende Antwortnachricht wird in der Regel ebenfalls geschützt übertragen.

Für die Längenbeschränkung von Kommando- und Antwortnachrichten gilt:

G2_N029.890.a.1 - (N029.890)a.1 K_COS
Wenn die Option_logische_Kanäle unterstützt wird, dann MUSS das COS ungeschützte Kommandonachrichten mit einer Länge aus dem Intervall [4, 1033] Oktett unterstützen.**[<=]**

A_15815 - (N029.890)a.2 K_COS
Wenn die Option_logische_Kanäle unterstützt wird, dann MUSS das COS ungeschützte Antwortnachrichten mit einer Länge aus dem Intervall [2, 32770] Oktett unterstützen.**[<=]**

G2_N029.890.a.3 - (N029.890)a.3 K_COS
Wenn die Option_logische_Kanäle unterstützt wird, dann MUSS das COS geschützte Kommandonachrichten mit einer Länge aus dem Intervall [16, 1033] Oktett unterstützen.**[<=]**

A_15816 - (N029.890)a.4 K_COS
Wenn die Option_logische_Kanäle unterstützt wird, dann MUSS das COS geschützte Antwortnachrichten mit einer Länge aus dem Intervall [16, 1033] Oktett unterstützen.**[<=]**

G2_N029.890.b.1 - (N029.890)b.1 K_COS
Wenn die Option_logische_Kanäle nicht unterstützt wird, dann MUSS das COS ungeschützte Kommandonachrichten mit einer Länge aus dem Intervall [4, 2057] Oktett unterstützen.**[<=]**

A_15817 - (N029.890)b.2 K_COS
Wenn die Option_logische_Kanäle nicht unterstützt wird, dann MUSS das COS ungeschützte Antwortnachrichten mit einer Länge aus dem Intervall [2, 32770] Oktett unterstützen.**[<=]**

G2_N029.890.b.3 - (N029.890)b.3 K_COS
Wenn die Option_logische_Kanäle nicht unterstützt wird, dann MUSS das COS geschützte Kommandonachrichten mit einer Länge aus dem Intervall [16, 2057] Oktett unterstützen.**[<=]**

A_15818 - (N029.890)b.4 K_COS
Wenn die Option_logische_Kanäle nicht unterstützt wird, dann MUSS das COS geschützte Antwortnachrichten mit einer Länge aus dem Intervall [16, 2057] Oktett unterstützen.**[<=]**

A_15819 - (N029.890)c K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für die Länge von Nachrichten
1. unterstützt oder
2. ablehnt.**[<=]**

A_15820 - (N029.892)a K_Anwendungsspezifikation {K_Karte}
Die Anwendungsspezifikation MUSS ein EF.ATR so spezifizieren, dass

4. es als Kindobjekt im Ordner *root* (siehe (N019.900)a) enthalten ist und
5. Längenbeschränkung sowohl für Kommando- und Antwortnachrichten als auch für geschützte und ungeschützte Nachrichten enthält, dabei gelten folgende Definitionen:
   a. *limitCmdPlain* ist die maximale Anzahl von Oktetten in einer ungeschützt übertragenen Kommandonachricht.
   b. *limitCmdSecureMessaging* ist die maximale Anzahl von Oktetten in einer geschützt übertragenen Kommandonachricht.
   c. *limitRspPlain* ist die maximale Anzahl von Oktetten in einer ungeschützt übertragenen Antwortnachricht.
   d. *limitRspSecureMessaging* ist die maximale Anzahl von Oktetten in einer geschützt übertragenen Antwortnachricht.

[<=]

A_15821 - (N029.892)b K_Karte
Der Hersteller MUSS die Längenangaben im EF.ATR so wählen, dass sie größer oder gleich den Längenbeschränkungen aus (N029.890) sind.**[<=]**

12 Kanalkontext (normativ)

Während das Objektsystem gemäß Kapitel 9 Informationen bündelt, die persistent in der Smartcard zu speichern sind und in allen logischen Kanälen gleichermaßen zur Verfügung stehen, wird in diesem Kapitel auf die Information eingegangen, die lediglich zwischen Öffnen und Schließen eines logischen Kanals zur Verfügung steht und damit kanalspezifisch ist. Die kanalspezifischen Attribute werden im Objekt *channelContext* zusammengefasst.

Die im Folgenden genannten Attribute werden typischerweise einem flüchtig (im RAM) gespeicherten Security Environment (siehe 8.8) oder Sicherheitsstatus (siehe 8.9) zugerechnet, oder durch das in 14.9.9.1 beschriebene Kommando MANAGE SECURITY ENVIRONMENT (restore) verändert.

12.1 Attribute eines logischen Kanals

An der physikalischen Schnittstelle (siehe CosA_e09) verhält sich das COS so, als gäbe es für jeden logischen Kanal ein Objekt *channelContext* mit den folgenden Attributen:

A_15825 - (N029.900) K_COS
Folgende Attribute von *channelContext* sind dem Objektsystem zugeordnet: Der *channelContext* MUSS **[<=]**

G2_N029.900.a - (N029.900)a K_COS
genau ein Attribut *currentFolder* enthalten, das auf ein Objekt vom Typ Ordner zeigt.**[<=]**

G2_N029.900.b - (N029.900)b K_COS
genau ein Attribut *RND.ICC* enthalten, welches eine vom COS erzeugte Zufallszahl oder den Wert "NoRandom" speichert (siehe (N086.902)a.8~~, b.8~~ und (N099.300)).**[<=]**

A_15826 - (N029.900)c K_COS
genau eine Liste *keyReferenceList* besitzen. Jedes Listenelement MUSS einen Wert unterstützen, der anzeigt, dass es leer ist. Die Liste *keyReferenceList* MUSS sich aus folgenden Elementen zusammensetzen:**[<=]**

G2_N029.900.c.1 - (N029.900)c.1 K_COS
externalAuthenticate, mit den Komponenten *keyReference* und *algorithmIdentifier*. **[<=]**

G2_N029.900.c.2 - (N029.900)c.2 K_COS
internalAuthenticate, mit den Komponenten *keyReference* und *algorithmIdentifier*. **[<=]**

G2_N029.900.c.3 - (N029.900)c.3 K_COS
verifyCertificate, mit der Komponente *keyReference*. **[<=]**

G2_N029.900.c.4 - (N029.900)c.4 K_COS
signatureCreation, mit den Komponenten *keyReference* und *algorithmIdentifier*. **[<=]**

G2_N029.900.c.5 - (N029.900)c.5 K_COS
dataDecipher, mit den Komponenten *keyReference* und *algorithmIdentifier*. **[<=]**

G2_N029.900.c.6 - (N029.900)c.6 K_COS
dataEncipher, mit den Komponenten *keyReference* und *algorithmIdentifier*. **[<=]**

G2_N029.900.c.7 - (N029.900)c.7 K_COS
macCalculation mit den Komponenten *keyReference* und *algorithmIdentifier*. **[<=]**

G2_N029.900.d - (N029.900)d.(1,2,3,4,5,6,7,8) K_COS
genau ein Attribut *SessionkeyContext* besitzen, welches folgende Elemente enthält:

1. *flagSessionEnabled* für welches folgender Wertebereich definiert ist:
   a. noSK zeigt an, dass keine Sessionkeys vorhanden sind.
   b. SK4SM zeigt an, dass die übrigen Attribute gebrauchsfertiges kryptographisches Material enthalten, welches im Layer SecMes (siehe CosA_e09) zum Entsichern einer Kommando-APDU oder zum Sichern einer Antwort-APDU verwendbar ist.
   c. SK4TC zeigt an, dass die übrigen Attribute gebrauchsfertiges kryptographisches Material enthalten, welches im Layer CmdInterpreter (siehe CosA_e09) verwendbar ist (beispielsweise im Rahmen von PSO-Kommandos).
2. *Kenc* ist ein symmetrischer Schlüssel zur Ver- und Entschlüsselung.
3. *Kmac* ist ein symmetrischer Schlüssel zur MAC-Berechnung und MAC-Verifikation.
4. Dieser Punkt ist absichtlich leer.~~*SSCenc* ist eine nicht-negative, ganze Zahl, die als Send Sequence Counter im Zusammenhang mit *Kenc* verwendet wird.~~
5. *SSCmac* ist eine nicht-negative, ganze Zahl, die als Send Sequence Counter im Zusammenhang mit *Kmac* verwendet wird.
6. *flagCmdEnc* ist eine boolesche Variable, welche anzeigt, ob die gesicherte Kommando-APDU ein Datenobjekt mit verschlüsselten Kommandodaten enthält. Dieses Flag wird in (N031.700)a.1 mit einem Wert versehen und in (N022.600)b ausgewertet.
7. *flagRspEnc* ist eine boolesche Variable, welche anzeigt, ob die Daten einer Antwort-APDU verschlüsselt übertragen werden. Das Flag wird in (N022.600)d mit einem Wert versehen und in~~(N033.700) bzw.~~ (N033.800) ausgewertet.
8. *negotiationKeyInformation* ist eine Variable, welche Informationen zum Schlüssel enthält, der an der Etablierung der Sessionkeys beteiligt war. Dabei sind folgende Fälle zu unterscheiden: *negotiationKeyInformation* enthält
   a. eine Schlüsselreferenz gemäß (N099.600) auf das beteiligte Authentisierungsobjekt, wenn dieses folgenden Typ besitzt:
      A. symmetrischem Authentisierungsobjekt (siehe 8.6.1).
      B. symmetrisches Kartenverbindungsobjekt (siehe 8.6.2).
   b. *accessRight* gemäß (N019.700) des beteiligten Schlüsselobjektes, wenn dieses folgenden Typ besitzt: öffentliches Authentisierungsobjekt (siehe 8.6.4.2).

[<=]

G2_N029.900.d.9 - (N029.900)d.9 K_COS, Option_Kryptobox
i. *accessRulesSessionkeys* ist eine Variable vom Typ *interfaceDependentAccessRules* (siehe 8.1.4), welche die Zugriffsregeln enthält, die im Rahmen einer Trusted-Channel-Unterstützung

ausgewertet werden, siehe (N087.244), (N090.200), (N091.650)b und (N096.364).

ii. *folderSessionkeys* ist eine Variable, die angibt, welchem Ordner die Sessionkeys zugeordnet sind.

[<=]

G2_N029.900.e.1 - (N029.900)e.1 K_COS
genau eine Liste *globalSecurityList* besitzen, wobei das COS für die Länge der Liste alle Werte aus dem Intervall [0, 3] unterstützen MUSS.**[<=]**

A_15827 - (N029.900)e.2 K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS bezüglich *globalSecurityList* längere Listen unterstützt.
[<=]

G2_N029.900.f.1 - (N029.900)f.1 K_COS
genau eine Liste *dfSpecificSecurityList* besitzen, wobei das COS für die Länge der Liste alle Werte aus dem Intervall [0, 3] unterstützen MUSS.**[<=]**

A_15828 - (N029.900)f.2 K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS bezüglich *dfSpecificSecurityList* längere Listen unterstützt.
[<=]

A_15829 - (N029.900)g K_COS
Jedes Element der Liste *globalSecurityList* und jedes Element der Liste *dfSpecificSecurityList* MUSS entweder**[<=]**

(N029.900)g.1 ist absichtlich leer.

G2_N029.900.g.1 - (N029.900)g.1 K_COS
~~Option_RSA_CVC, ein *CHA* gemäß (N019.700)a sein in Verbindung mit einer Referenz zu einem Ordner, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß (N084.400)b oder (N084.400)c stattgefunden hat mit einem Schlüsselobjekt aus vorgenanntem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d),~~**[<=]**

G2_N029.900.g.2 - (N029.900)g.2 K_COS
eine Referenz auf ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) sein, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß (N084.402)a~~, b~~ oder (N084.410)a~~, b~~ stattgefunden hat, oder**[<=]**

G2_N029.900.g.3 - (N029.900)g.3 K_COS Option_kontaktlose_Schnittstelle
eine Referenz auf ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) sein, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß 15.4.2 stattgefunden hat.**[<=]**

G2_N029.900.h.1 - (N029.900)h.1 K_COS
genau eine Liste *bitSecurityList* besitzen, wobei das COS für die Länge der Liste alle Werte aus dem Intervall [0, 1] unterstützen MUSS.**[<=]**

A_15830 - (N029.900)h.2 K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS bezüglich *bitSecurityList* längere Listen unterstützt.**[<=]**

G2_N029.900.h.3 - (N029.900)h.3 K_COS

Jedes Element der Liste *bitSecurityList* MUSS ein CHAT sein in Verbindung mit einer Referenz zu einem Ordner, wodurch angezeigt wird, dass eine erfolgreiche Komponentenauthentisierung gemäß (N084.400)a oder (N085.054) oder (N085.056) stattgefunden hat mit einem Schlüsselobjekt aus vorgenanntem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d.1).**[<=]**

G2_N029.900.i.1 - (N029.900)i.1 K_COS
genau eine Liste *globalPasswordList* besitzen, wobei das COS für die Länge der Liste alle Werte aus dem Intervall [0, 4] unterstützen MUSS.**[<=]**

A_15831 - (N029.900)i.2 K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS bezüglich *globalPasswordList* längere Listen unterstützt.**[<=]**

G2_N029.900.j.1 - (N029.900)j.1 K_COS
genau eine Liste *dfSpecificPasswordList* besitzen, wobei das COS für die Länge der Liste alle Werte aus dem Intervall [0, 4] unterstützen MUSS.**[<=]**

A_15832 - (N029.900)j.2 K_TST
~~Ein COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS bezüglich *dfSpecificPasswordList* längere Listen unterstützt.**[<=]**

G2_N029.900.k - (N029.900)k K_COS
Jedes Element der Liste *globalPasswordList* und jedes Element der Liste *dfSpecificPasswordList* MUSS genau ein Attribut *securityStatusEvaluationCounter* sein in Verbindung mit einer Referenz auf ein Passwortobjekt, wodurch angezeigt wird, dass eine erfolgreiche Benutzerverifikation gemäß (N082.200) mit diesem Passwortobjekt stattgefunden hat.**[<=]**

G2_N029.900.l.1 - (N029.900)l.1 K_COS
Der Wertebereich von *securityStatusEvaluationCounter* MUSS alle Werte von *startSsec* (siehe (N015.800)c.2) umfassen.**[<=]**

A_15834 - (N029.900)l.2 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS bezüglich *securityStatusEvaluationCounter* weitere Werte
i. unterstützt, oder
ii. ablehnt. **[<=]**

G2_N029.900.m - (N029.900)m K_COS
genau ein Attribut *currentEF* enthalten, das
1. entweder unbestimmt ist,
2. oder auf ein Listenelement von *currentFolder.children* vom Typ Datei zeigt.**[<=]**

A_15835 - (N030.000) K_COS
Folgende Attribute von *channelContext* sind einem Ordner zugeordnet:**[<=]**

G2_N030.000.a - (N030.000)a K_COS
Der *channelContext* MUSS für jeden Ordner im Objektsystem genau ein Attribut *seIdentifier* gemäß (N007.900) enthalten.**[<=]**

G2_N030.010 - (N030.010) K_COS, Option_logische_Kanäle
Das COS MUSS mindestens vier logische Kanäle unterstützen. Das heißt neben dem Basiskanal sind mindestens drei weitere logische Kanäle zu unterstützen.**[<=]**

12.2 Reset-Verhalten

Ein "neuer" Kanalkontext gemäß12.1 wird dann etabliert, wenn ein Reset durchgeführt wird, oder ein neuer logischer Kanal geöffnet wird. Dann gilt:

A_15838 - (N030.000) K_COS
Wenn der logische Kanal mit der Kanalnummer null (Basiskanal) durch**[<=]**

A_17516 - (N030.100)a K_COS
Einschalten des physikalischen Interfaces, das heißt**[<=]**

G2_N030.100.a.1 - (N030.100)a.1 K_COS
für das Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11] wird eine Aktivierung gemäß (N023.920)a durchgeführt,**[<=]**

G2_N030.100.a.2 - (N030.100)a.2 K_COS, Option_USB_Schnittstelle
für das USB-Übertragungsprotokoll gemäß 11.2.2 wird eine Aktivierung gemäß [ISO/IEC 7816-12] durchgeführt,**[<=]**

G2_N030.100.a.3 - (N030.100)a.3 K_COS, Option_kontaktlose_Schnittstelle
für die kontaktlose Datenübertragung gemäß 11.2.3 wird eine Aktivierung gemäß ISO/IEC 14443 durchgeführt.**[<=]**

G2_N030.100.b - (N030.100)b K_COS
durch einen Warm-Reset gemäß (N023.920)c**[<=]**

geöffnet wird oder ein anderer logischer Kanal durch das Kommando MANAGE CHANNEL geöffnet wird (siehe (N099.508)), oder ein logischer Kanal zurückgesetzt wird (siehe (N099.524) und (N099.532)) dann MUSS für den betroffenen Kanal gelten:

G2_N030.100.c - (N030.100)c K_COS
Dem betroffenen logischen Kanal MUSS exklusiv ein Kanalkontext gemäß 12.1 zugeordnet werden.**[<=]**

G2_N030.100.d - (N030.100)d K_COS
Das Attribut *currentFolder* MUSS auf *root* gesetzt werden.**[<=]**

G2_N030.100.e - (N030.100)e K_COS
Das Attribut *RND.ICC* MUSS auf den Wert "NoRandom" gesetzt werden.**[<=]**

G2_N030.100.f - (N030.100)f K_COS
Das Attribut *keyReferenceList* MUSS so gesetzt werden, dass alle Elemente leer sind.**[<=]**

G2_N030.100.g - (N030.100)g K_COS
Das Attribut *SessionkeyContext.flagSessionEnabled* MUSS auf noSK gesetzt werden.**[<=]**

G2_N030.100.h - (N030.100)h K_COS
Die Liste *globalSecurityList* MUSS leer sein.**[<=]**

G2_N030.100.i - (N030.100)i K_COS
Die Liste *dfSpecificSecurityList* MUSS leer sein.**[<=]**

G2_N030.100.j - (N030.100)j K_COS
Das Attribut *bitSecurityList* MUSS leer sein.**[<=]**

G2_N030.100.k - (N030.100)k K_COS
Die Liste *globalPasswordList* MUSS leer sein.**[<=]**

G2_N030.100.l - (N030.100)l K_COS

Die Liste *dfSpecificPasswordList* MUSS leer sein.**[<=]**

G2_N030.100.m.1 - (N030.100)m.1 K_COS
Für alle Ordner gilt: *seIdentifier* MUSS auf den Wert eins gesetzt werden.**[<=]**

G2_N030.100.n - (N030.100)n K_COS
Das Attribut *currentEF* MUSS auf den Wert "unbestimmt" gesetzt werden.**[<=]**

Hinweis CosH_3bf: In (N030.100) sind absichtlich keine Zustände oder Zustandsübergänge aus ISO/IEC 14443 aufgeführt. Daraus folgt, dass mit Ausnahme des Zustandsübergangs nach "POWER-OFF" kein Kanalkontext durch Zustandsübergänge beeinflusst wird.

12.3 Setzen eines Sicherheitsstatus

Die hier beschriebene Routine setzt den Sicherheitsstatus des als Parameter übergebenen Authentisierungsschlüssels.

Tabelle 84: CosT_e4d: Definition der Funktion setSecurityStatus(...)

Input:	*obj*	Ein Schlüsselobjekt
Output:	–	Kein Rückgabewert
Notation:		setSecurityStatus(*obj*)

G2_N030.200.a - (N030.200)a K_COS
Wenn *obj* im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalSecurityList* gelten (siehe (N029.900)e).**[<=]**

G2_N030.200.b - (N030.200)b K_COS
Wenn *obj* einem von *root* (siehe (N019.900)a) verschiedenen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificSecurityList* gelten (siehe (N029.900)f).**[<=]**

G2_N030.300.a.1 - (N030.300)a.1 K_COS
Wenn *obj* ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) und *obj* in der Liste *tmpList* bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.**[<=]**

G2_N030.300.a.2 - (N030.300)a.2 K_COS
Wenn *obj* ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) und *obj* in der Liste *tmpList* noch nicht vorhanden ist, dann MUSS *obj* am Anfang von *tmpList* eingetragen werden.**[<=]**

(N030.300)b ist absichtlich leer.

G2_N030.300.b.1 - (N030.300)b.1 K_COS, Option_RSA_CVC
~~Wenn *obj* ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein RSA-Schlüssel ist und *obj.CHA* (siehe (N019.700)a) in der Liste *tmpList* bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.~~**[<=]**

G2_N030.300.b.2 - (N030.300)b.2 K_COS, Option_RSA_CVC
~~Wenn *obj* ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein RSA-Schlüssel ist und *obj.CHA* (siehe (N019.700)a) in der Liste *tmpList* noch nicht vorhanden ist, dann MUSS *obj.CHA* am Anfang von *tmpList* eingetragen werden zusammen mit einem Verweis auf den Ordner, der auch *obj* enthält. Dadurch wird der Sicherheitsstatus von *obj.CHA* demselben Ordner zugeordnet, der auch *obj* enthält.~~ **[<=]**

G2_N030.300.c - (N030.300)c K_COS
Wenn *tmpList* durch Eintragungen länger wurde, als vom COS unterstützt, dann MUSS das COS das letzte Listenelement (hier als *objLast* bezeichnet) mittels clearSecurityStatusKey(*objLast*) entfernen (FIFO = first in first out).**[<=]**

A_15839 - (N030.400) K_COS
Wenn *obj* ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein ELC-Schlüssel ist und *obj*.CHAT (siehe (N019.700)b.1) in der Liste *bitSecurityList* (siehe (N029.900)h)**[<=]**

G2_N030.400.a - (N030.400)a K_COS
bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.**[<=]**

G2_N030.400.b - (N030.400)b K_COS
noch nicht vorhanden ist, dann MUSS *obj*.CHAT am Anfang von *bitSecurityList* eingetragen werden zusammen mit einem Verweis auf den Ordner, der auch *obj* enthält. Dadurch wird der Sicherheitsstatus von *obj*.CHAT demselben Ordner zugeordnet, der auch *obj* enthält. **[<=]**

G2_N030.400.c - (N030.400)c K_COS
Wenn *bitSecurityList* durch Eintragungen länger wurde, als vom COS unterstützt, dann MUSS das COS das letzte Listenelement (hier als *objLast* bezeichnet) mittels clearSecurityStatusKey(*objLast*) entfernen (FIFO = first in first out).**[<=]**

12.4 Löschen eines Sicherheitsstatus

12.4.1 Löschen des Sicherheitszustandes eines Schlüssels

Die hier beschriebene Routine löscht den Sicherheitsstatus des als Parameter übergebenen Authentisierungsschlüssels.

Tabelle 85: CosT_489: Definition der Funktion clearSecurityStatusKey(...) für Authentisierungsschlüssel

Input:	*obj*	Ein Schlüsselobjekt
Output:	–	Kein Rückgabewert
Notation:		clearSecurityStatusKey(*obj*)

G2_N030.500.a - (N030.500)a K_COS
Wenn *obj* im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalSecurityList* gelten (siehe (N029.900)e).**[<=]**

G2_N030.500.b - (N030.500)b K_COS
Wenn *obj* einem von *root* (siehe (N019.900)a) verschiedenen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificSecurityList* gelten (siehe (N029.900)f).**[<=]**

G2_N030.600.a.1 - (N030.600)a.1 K_COS
Wenn *obj* ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) und *obj* in der Liste *tmpList* nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.**[<=]**

G2_N030.600.a.2 - (N030.600)a.2 K_COS

Wenn *obj* ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) und *obj* in der Liste *tmpList* noch vorhanden ist, dann MUSS *obj* aus *tmpList* entfernt werden.**[<=]**

(N030.600)b.1 ist absichtlich leer.

G2_N030.600.b.1.i - (N030.600)b.1.i K_COS, Option_RSA_CVC
~~RSA-Schlüssel ist und *obj.CHA* (siehe (N019.700)a) in der Liste *tmpList* nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.~~**[<=]**

G2_N030.600.b.1.ii - (N030.600)b.1.ii K_COS, Option_RSA_CVC
~~RSA-Schlüssel ist und *obj.CHA* (siehe (N019.700)a) in der Liste *tmpList* noch vorhanden ist, dann MUSS *obj.CHA* aus *tmpList* entfernt werden.~~**[<=]**

G2_N030.600.b.2.i - (N030.600)b.2.i K_COS
Wenn *obj* ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein ELC-Schlüssel ist und *obj*.CHAT (siehe (N019.700)b.1) in der Liste *bitSecurityList* (siehe (N029.900)h) nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.**[<=]**

G2_N030.600.b.2.ii - (N030.600)b.2.ii K_COS
Wenn *obj* ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein ELC-Schlüssel ist und *obj*.CHAT (siehe (N019.700)b.1) in der Liste *bitSecurityList* (siehe (N029.900)h) noch vorhanden ist, dann MUSS *obj*.CHAT aus *bitSecurityList* entfernt werden.**[<=]**

G2_N030.600.c - (N030.600)c K_COS
Wenn *obj* einen anderen Typ besitzt, dann MUSS dieser Algorithmus beendet werden.**[<=]**

G2_N030.700 - (N030.700) K_COS
Wenn der Sicherheitszustand eines Schlüssels gelöscht wurde, der an der Aushandlung von Sessionkeys beteiligt war, dann MUSS *SessionkeyContext.flagSessionEnabled* (siehe (N029.900)d.1) auf den Wert noSK gesetzt werden.**[<=]**

12.4.2 Löschen der Sicherheitszustände eines Ordners

Die hier beschriebene Routine löscht die Sicherheitszustände aller Schlüssel, welche dem als Parameter übergebenen Ordner zugeordnet sind.

Tabelle 86: CosT_956: Definition der Funktion clearSecurityStatus(...) für Ordner

Input:	*obj*	Ein Ordner (siehe 8.3.1)
Output:	–	Kein Rückgabewert
Notation:		clearSecurityStatusFolder(*obj*)

G2_N030.720 - (N030.720) K_COS
Die dem Ordner *obj* zugeordneten Sicherheitszustände von Schlüsseln sind wie folgt zu löschen: Jedes Element in *globalSecurityList* (siehe (N029.900)e) und in *dfSpecificSecurityList* (siehe (N029.900)f) und in *bitSecurityList* (siehe (N029.900)h) MUSS mittels clearSecurityStatusKey(…) aus den genannten Listen entfernt werden, falls es *obj* zugeordnet ist.**[<=]**

12.4.3 Löschen von Sessionkeys

Die hier beschriebene Routine löscht den Sicherheitsstatus, welcher bei der Aushandlung von Sessionkeys gesetzt wurde.

Tabelle 87: CosT_e7c: Definition der Funktion clearSessionkeys()

Input:	–	Kein Parameter
Output:	–	Kein Rückgabewert
Notation:		clearSessionkeys( )

Der Sicherheitszustand, welcher bei der Aushandlung von Sessionkeys gesetzt wurde, ist wie folgt zu löschen:

G2_N030.740.a - (N030.740)a K_COS
Das Elementen in *globalSecurityList* (siehe (N029.900)e) oder *dfSpecificSecurityList* (siehe (N029.900)f) oder *bitSecurityList* (siehe (N029.900)h) MUSS aus den genannten Listen entfernt werden, welches an der Aushandlung der Sessionkeys beteiligt war.**[<=]**

G2_N030.740.b - (N030.740)b K_COS
SessionkeyContext.flagSessionEnabled (siehe (N029.900)d.1) MUSS auf den Wert noSK gesetzt werden.**[<=]**

12.5 Setzen eines Passwortstatus

Die hier beschriebene Routine setzt den Sicherheitsstatus des als Parameter übergebenen Passwortes.

Tabelle 88: CosT_1ba: Definition der Funktion setPasswordStatus(...)

Input:	*obj*	Ein Passowortobjekt
Output:	–	Kein Rückgabewert
Notation:		setPasswordStatus(*obj*)

G2_N030.800.a - (N030.800)a K_COS
Wenn *obj* im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalPasswordList* gelten (siehe (N029.900)i).**[<=]**

G2_N030.800.b - (N030.800)b K_COS
Wenn *obj* einem von *root* (siehe (N019.900)a) verschiedenen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificPasswordList* gelten (siehe (N029.900)j).**[<=]**

G2_N030.900.a - (N030.900)a K_COS
Wenn *obj* in der Liste *tmpList* bereits vorhanden ist, dann MUSS zunächst clearPasswordStatus(*obj*) und dann (N030.900)b ausgeführt werden.**[<=]**

G2_N030.900.b - (N030.900)b K_COS
Wenn *obj* in der Liste *tmpList* noch nicht vorhanden ist, dann MUSS *obj* am Anfang von *tmpList* eingetragen werden. Dabei MUSS das Attribut *securityStatusEvaluationCounter* des neuen Listenelementes in *tmpList* (siehe (N029.900)k) auf den Wert *startSsec* (siehe (N015.800)c.2) gesetzt werden, der aus *obj.startSsecList* (siehe (N015.800)) unter

Berücksichtigung von *seIdentifier* aus (N015.800)c.1 und *seIdentifier* aus (N030.000)a ermittelt wird. Dabei MUSS in (N030.000) der Ordner zu Grunde gelegt werden, der *obj* enthält. **[<=]**

G2_N031.000 - (N031.000) K_COS
Wenn *tmpList* durch Eintragungen länger wurde, als vom COS unterstützt, dann MUSS das COS das letzte Listenelement entfernen (FIFO = first in first out).**[<=]**

G2_N031.100 - (N031.100) K_COS
Im Eintrag zu *obj* in *tmpList* MUSS das Attribut *securityStatusEvaluationCounter* auf den Wert *obj.startSsec* gesetzt werden.**[<=]**

12.6 Löschen eines Passwortstatus

Die hier beschriebene Routine löscht den Sicherheitsstatus des als Parameter übergebenen Passwortes.

Tabelle 89: CosT_f93: Definition der Funktion clearPasswordStatus(...)

Input:	*obj*	Ein Passwortobjekt
Output:	–	Kein Rückgabewert
Notation:		clearPasswordStatus(*obj*)

G2_N031.200.a - (N031.200)a K_COS
Wenn *obj* im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalPasswordList* gelten (siehe (N029.900)i).**[<=]**

G2_N031.200.b - (N031.200)b K_COS
Wenn *obj* einem von *root* (siehe (N019.900)a) verschiedenen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificPasswordList* gelten (siehe (N029.900)j).**[<=]**

G2_N031.300.a - (N031.300)a K_COS
Wenn *obj* in der Liste *tmpList* nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.**[<=]**

G2_N031.300.b - (N031.300)b K_COS
Wenn *obj* in der Liste *tmpList* noch vorhanden ist, dann MUSS *obj* aus *tmpList* entfernt werden.**[<=]**

13 Gesicherte Kommunikation (normativ)

13.1 Secure Messaging Layer

Dieses Unterkapitel beschreibt die Funktionsweise des Layers "SecMes" in CosA_e09. Dieser Layer benutzt neben den Informationen aus (N029.900)d folgende weitere Attribute:

Tabelle 90: CosT_a2e: Definition von Oktettstring zur Schlüsselableitung

KD.i	ein Oktettstring, der eine typischerweise vom COS generierte Zufallszahl speichert, die im Rahmen der Ableitung von Sessionkeys verwendet wird
KD.e	ein Oktettstring, der eine extern generierte Zufallszahl speichert, die im Rahmen der Ableitung von Sessionkeys verwendet wird

13.1.1 Ableitung von Sessionkeys

Tabelle 91: CosT_b83: Definition der Funktion SessionkeyDerivation()

Input:	–	Der benötigte Input wird dem *channelContext* entnommen
Output:	–	Kein Output vorhanden
Errors:	–	Keine
Notation:		SessionkeyDerivation()

Im Folgenden gelten die Definitionen:

Tabelle 92: CosT_d73: Definitionen im Rahmen der Ableitung von Sessionkeys

algId	=	*channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* oder *channelContext.keyReferenceList.internalAuthenticate.algorithmIdentifier*
PrK	=	an der Authentisierung beteiligter privater Schlüssel, referenziert in *channelContext.keyReferenceList.internalAuthenticate.keyReference*
PuK	=	an der Authentisierung beteiligter öffentlicher Schlüssel
SK	=	an der Authentisierung beteiligter symmetrischer Schlüssel, referenziert in *channelContext.keyReferenceList.externalAuthenticate.keyReference*

Hinweis CosH_e5c: Bei der Aushandlung von Sessionkeys sind zwei Fälle zu unterscheiden: Entweder die Sessionkey-Aushandlung erfolgte mittels

a. *symmetrischer Schlüssel, dann ist aus den obigen Definitionen* SK *relevant und sowohl* PrK *als auch* PuK *sind irrelevant, oder*

b. *die Sessionkey-Aushandlung erfolgte mittels asymmetrischer Schlüssel, dann ist aus den obigen Definitionen* SK *irrelevant und sowohl* PrK *als auch* PuK *sind relevant.*

Das Attribut *SessionkeyContext. negotiationKeyInformation* wird wie folgt gesetzt:

G2_N031.390.a - (N031.390)a K_COS
Wenn die Sessionkeys asymmetrisch ausgehandelt wurden, dann MUSS gelten:
SessionkeyContext. negotiationKeyInformation = *PuK. accessRights*. **[<=]**

G2_N031.390.b - (N031.390)b K_COS
Wenn die Sessionkeys symmetrisch ausgehandelt wurden, dann MUSS gelten:
SessionkeyContext. negotiationKeyInformation = *channelContext.keyReferenceList.externalAuthenticate.keyReference*. **[<=]**

(N031.400) ist absichtlich leer.

A_15844 - (N031.400) K_COS, Option_DES
~~Wenn der an der Aushandlung beteiligte Aushandlungsschlüssel die Aushandlung von 3TDES-Schlüsseln impliziert, dann MUSS für die Attribute aus *SessionkeyContext* gelten:~~**[<=]**

G2_N031.400.a - (N031.400)a K_COS, Option_DES
~~(*Kenc*, *SSCenc*, *Kmac*, *SSCmac*) = KeyDerivation_3TDES(KD.i XOR KD.e).~~**[<=]**

G2_N031.400.b - (N031.400)b K_COS, Option_DES
~~Für *algIDSessionkey* in (N031.522) MUSS dann *algIDSessionkey* = desSessionkey gelten.~~**[<=]**

Wenn der an der Aushandlung beteiligte Aushandlungsschlüssel die Aushandlung von AES-Schlüsseln impliziert, dann gilt für die Attribute aus *SessionkeyContext*:

G2_N031.500.a.1 - (N031.500)a.1 K_COS
Wenn die Sessionkeys symmetrisch ausgehandelt wurden und *SK* ein AES-128 Schlüssel ist, dann MUSS gelten:
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES128(KD.i XOR KD.e).**[<=]**

G2_N031.500.a.2 - (N031.500)a.2 K_COS
Wenn die Sessionkeys symmetrisch ausgehandelt wurden und *SK* ein AES-192 Schlüssel ist, dann MUSS gelten:
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES192(KD.i XOR KD.e).**[<=]**

G2_N031.500.a.3 - (N031.500)a.3 K_COS
Wenn die Sessionkeys symmetrisch ausgehandelt wurden und *SK* ein AES-256 Schlüssel ist, dann MUSS gelten:
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES256KD.i XOR KD.e).**[<=]**

G2_N031.500.b.1 - (N031.500)b.1 K_COS
Wenn die Sessionkeys asymmetrisch ausgehandelt wurden und *PrK.domainParameter* gleich brainpoolP256r1 ist, dann MUSS gelten:
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES128(KD.i XOR KD.e).**[<=]**

G2_N031.500.b.2 - (N031.500)b.2 K_COS
Wenn die Sessionkeys asymmetrisch ausgehandelt wurden und *PrK.domainParameter* gleich brainpoolP384r1 ist, dann MUSS gelten:
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES192(KD.i XOR KD.e).**[<=]**

G2_N031.500.b.3 - (N031.500)b.3 K_COS
Wenn die Sessionkeys asymmetrisch ausgehandelt wurden und *PrK.domainParameter* gleich brainpoolP512r1 ist, dann MUSS gelten:
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES256KD.i XOR KD.e).**[<=]**

G2_N031.500.c.1 - (N031.500)c.1 K_COS
Wenn die Sessionkeys mittels PACE ausgehandelt wurden und *channelContext.keyReferenceList.externalAuthenticate.algID* ist Element der Menge, dann MUSS gelten:
{id-PACE-ECDH-GM-AES-CBC-CMAC-128, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128}
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES128(KD.i XOR KD.e).**[<=]**

G2_N031.500.c.2 - (N031.500)c.2 K_COS
Wenn die Sessionkeys mittels PACE ausgehandelt wurden und *channelContext.keyReferenceList.externalAuthenticate.algID* ist Element der Menge, dann MUSS gelten:
{id-PACE-ECDH-GM-AES-CBC-CMAC-192, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192}
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES192(KD.i XOR KD.e).**[<=]**

G2_N031.500.c.3 - (N031.500)c.3 K_COS
Wenn die Sessionkeys mittels PACE ausgehandelt wurden und *channelContext.keyReferenceList.externalAuthenticate.algID* ist Element der Menge, dann MUSS gelten:
{id-PACE-ECDH-GM-AES-CBC-CMAC-256, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256}
(*Kenc*, *Kmac*, *SSCmac*) = KeyDerivation_AES256KD.i XOR KD.e).**[<=]**

G2_N031.500.d - (N031.500)d K_COS
Für *algIDSessionkey* in (N031.522) MUSS dann *algIDSessionkey* = aesSessionkey gelten.**[<=]**

A_15847 - (N031.520) K_COS
Wenn *algId* anzeigt, dass die Sessionkeys im Rahmen von Secure Messaging einzusetzen sind, dann MUSS gesetzt werden**[<=]**

G2_N031.520.a - (N031.520)a K_COS
SessionkeyContext.flagSessionEnabled = SK4SM und**[<=]**

A_15848 - (N031.520)b K_COS
SessionkeyContext. accessRulesSessionkeys auf einen beliebigen Wert, weil die Zugriffsregeln der Sessionkeys im Rahmen dieser Spezifikation nicht ausgewertet werden (vergleiche auch (N031.522)b).**[<=]**

A_15849 - (N031.522) K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *algId* anzeigt, dass die Sessionkeys den Betrieb eines Trusted Channels zu unterstützen haben, dann MÜSSEN mit *algIDSessionkey* aus ~~(N031.400)b bzw.~~ (N031.500)d folgende Änderungen am *channelContext* vorgenommen werden:**[<=]**

G2_N031.522.a - (N031.522)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Setze *SessionkeyContext.flagSessionEnabled* = SK4TC.**[<=]**

G2_N031.522.b.1 - (N031.522)b.1 K_COS, Option_Kryptobox oder Option_PACE_PCD
Setze *SessionkeyContext. accessRulesSessionkeys* gleich *SK. accessRulesSessionkeys* falls die Sessionkeys symmetrisch vereinbart wurden.**[<=]**

G2_N031.522.b.2 - (N031.522)b.2 K_COS, Option_Kryptobox oder Option_PACE_PCD

Setze *SessionkeyContext. accessRulesSessionkeys* gleich *PrK. accessRulesSessionkeys*, falls die Sessionkeys asymmetrisch vereinbart wurden.**[<=]**

G2_N031.522.c - (N031.522)c K_COS, Option_Kryptobox oder Option_PACE_PCD
Setze *SessionkeyContext. folderSessionkeys* so, dass dadurch die Sessionkeys demselben Ordner zugeordnet werden wie *SK* bzw. *PrK*. **[<=]**

G2_N031.522.d - (N031.522)d K_COS, Option_Kryptobox oder Option_PACE_PCD
Trage in *channelContext.keyReferenceList.dataDecipher* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein.**[<=]**

G2_N031.522.e - (N031.522)e K_COS, Option_Kryptobox oder Option_PACE_PCD
Trage in *channelContext.keyReferenceList.dataEncipher* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein.**[<=]**

G2_N031.522.f - (N031.522)f K_COS, Option_Kryptobox oder Option_PACE_PCD
Trage in *channelContext.keyReferenceList.macCalculation* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein.**[<=]**

13.1.2 Bearbeitung einer Kommando-APDU

Dieses Kapitel beschreibt die generische Bearbeitung eines Kommandos. Hier, wie auch in Kapitel 14 werden gelegentlich kanalspezifische Angaben benötigt, wie etwa *currentFolder*. Auf eine spezielle Kennzeichnung, dass damit der Wert aus dem Kanalkontext *channelContext* des logischen Kanals zu nehmen ist, der im CLA-Byte angezeigt wird, wird der Übersichtlichkeit halber verzichtet.

A_15855 - (N031.600)a K_COS
Wenn im CLA-Byte kein Secure Messaging angezeigt wird (siehe [ISO/IEC 7816-4#5.4.1]) und *SessionkeyContext.flagSessionEnabled* den Wert SK4SM besitzt, dann MUSS das COS folgende zwei Schritte ausführen:**[<=]**

G2_N031.600.a.1 - (N031.600)a.1 K_COS
Schritt 1: Das COS MUSS den Sicherheitszustand des zugehörigen Aushandlungsschlüssels mittels clearSessionkeys() zurücksetzen (siehe CosT_e7c).**[<=]**

G2_N031.600.a.2 - (N031.600)a.2 K_COS
Schritt 2: Das COS MUSS CmdApdu3 = CmdApdu2 setzen.**[<=]**

G2_N031.600.b - (N031.600)b K_COS
Wenn im CLA-Byte kein Secure Messaging angezeigt wird (siehe [ISO/IEC 7816-4#5.4.1]) und *SessionkeyContext.flagSessionEnabled* einen anderen Wert als SK4SM besitzt, dann MUSS CmdApdu3 = CmdApdu2 gesetzt werden.**[<=]**

A_15856 - (N031.700)a K_COS
Wenn im CLA-Byte Secure Messaging angezeigt wird und *SessionkeyContext.flagSessionEnabled* besitzt den Wert SK4SM, dann MUSS das COS folgende zwei Schritte ausführen:**[<=]**

A_15857 - (N031.700)a.1.i K_externeWelt {K_Karte}
Die externe Welt MUSS die gesicherte Kommando-APDU CmdApdu2 gemäß den Vorgaben aus 13.2 codieren. **[<=]**

G2_N031.700.a.1 - (N031.700)a.1.ii K_COS
Schritt 1: Wenn ein DO mit Tag = '87' in der CmdApdu2 vorhanden ist, dann MUSS *flagCmdEnc* auf True gesetzt werden, sonst auf False.**[<=]**

A_15858 - (N031.700)a.1.iii K_TST
~~Das COS KANN weitere~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Secure Messaging Formate
A. unterstützt oder
B. ablehnt.**[<=]**

G2_N031.700.a.2 - (N031.700)a.2 K_COS
Schritt 2: Das COS MUSS CmdApdu3 mittels der Umkehroperation zu den Regeln in 13.2 aus CmdApdu2 berechnen. Wenn dabei ein Fehler festgestellt wird, genau dann

i. MUSS der Sicherheitszustand des zugehörigen Aushandlungsschlüssels mittels clearSessionkeys() zurückgesetzt werden (siehe CosT_e7c)
ii. MUSS die Bearbeitung des Kommandos terminieren.
iii. RspApdu2 enthält in diesem Fall keine Antwortdaten und besteht lediglich aus dem Trailer IncorrectSmDo = '69 88'.

[<=]

G2_N031.700.b - (N031.700)b K_COS
Wenn im CLA-Byte Secure Messaging angezeigt wird und *SessionkeyContext.flagSessionEnabled* einen anderen Wert als SK4SM, besitzt, dann MUSS

1. die Bearbeitung des Kommandos terminieren.
2. RspApdu2 DARF in diesem Fall NICHT mehr als den Trailer IncorrectSmDo = '69 88' enthalten. Das bedeutet, dass RspApdu2 in diesem Fall keine Antwortdaten enthält.

[<=]

G2_N031.800 - (N031.800) K_COS
Das COS MUSS die vom Secure Messaging Layer ("SecMes" in CosA_e09) gegebenenfalls weitergeleitete CmdApdu3 gemäß den Vorgaben aus Kapitel 14 bearbeiten, wobei eine Antwortnachricht-RspApdu1 entsteht.**[<=]**

Hinweis CosH_46d: In Kapitel 14 wird davon ausgegangen, dass der Secure Messaging Layer CmdApdu3 so aufbereitet, dass im CLA-Byte kein Secure Messaging angezeigt wird und die Kanalnummer auf null gesetzt wird.

G2_N031.900.a - (N031.900)a K_COS
Wenn *SessionkeyContext.flagSessionEnabled* den Wert SK4SM besitzt, dann MUSS die ungesicherte RspApdu1 gemäß den Vorgaben aus 13.3 in die gesicherte RspApdu2 umgewandelt werden.**[<=]**

G2_N031.900.b - (N031.900)b K_COS
Wenn *SessionkeyContext.flagSessionEnabled* einen anderen Wert als SK4SM besitzt, dann MUSS RspApdu2 = RspApdu1 gesetzt werden.**[<=]**

G2_N031.920 - (N031.920) K_COS
Wenn sowohl KD.i, als auch KD.e vorhanden sind (einer oder beide Werte wurden mit dem gerade bearbeiteten Kommando gesetzt), dann
a. MUSS *SessionkeyContext* mittels SessionkeyDerivation() geändert werden (siehe CosT_b83) und anschließend
b. MÜSSEN KD.i und KD.e auf den Wert "nicht vorhanden" gesetzt werden.**[<=]**

Während der Kommandobearbeitung ist es möglich, dass Fälle eintreten, die eine weitere Bearbeitung von Kommandos verhindern. Dieser Zustand ist wie folgt gekennzeichnet:

Die Kommando-APDU CmdApdu1 in CosA_e09 wird nicht mit einer korrespondierenden Antwort-APDU RspApdu3 beantwortet. Gemäß den in Kapitl 11 beschriebenen Übertragungsprotokollen ist es dann nicht möglich, eine weitere Kommando-APDU zu schicken.

A_15859 - (N031.940)a K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass der Prüfling in diesem Zustand WTX-Requests über "Interface physical" in CosA_e09 sendet.**[<=]**

A_15860 - (N031.940)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass der Prüfling in diesem Zustand keine WTX-Requests über "Interface physical" in CosA_e09 sendet.**[<=]**

A_15861 - (N031.940)c K_COS
Das COS MUSS diesen Zustand durch einen Reset verlassen.**[<=]**

Hinweis CosH_095: Wenn zum Zeitpunkt des Resets eine Transaktion offen ist (siehe 14.1), dann ist es möglich, dass im Rahmen eines Roll-Forward oder eines Roll-Back Fälle eintreten, die eine weitere Bearbeitung von Kommandos verhindern.

13.2 Sicherung einer Kommando-APDU

Dieses Unterkapitel beschreibt, wie eine gemäß 11.5 strukturierte Kommando-APDU zu sichern ist. Die hier beschriebenen Regeln richten sich an die Instanz, welche Kommando-APDU sendet. Das COS führt die entsprechenden Umkehroperationen durch.

Generell wird hier ein Subset der Regeln aus [ISO/IEC 7816-4#10] beschrieben, wobei

- alle Kommando-APDU mit Integritätsschutz übertragen werden,
- der Kommandoheader stets mit Integritätsschutz übertragen wird,
- Kommandodaten als Klartext oder verschlüsselt übertragen werden.

Hier sei angemerkt, dass der Sender zwar die Wahl hat zwischen der Übertragung von Kommandodaten im Klartext oder als Chiffrat, aber die Zugriffsregel gegebenenfalls eine verschlüsselte Übertragung erzwingt.

Wie in 11.5 beschrieben, besteht eine Kommando-APDU generisch betrachtet aus den Oktetten CLA, INS, P1 und P2 sowie aus dem optionalen Datenfeld Data und dem optionalen LeFeld. Somit gelten folgende Definitionen:

Tabelle 93: CosT_281: Definition der Funktionsparameter zur Sicherung einer Kommando-APDU

Input:	CLA	Oktett gemäß (N026.500)
	INS	Oktett gemäß (N026.600)
	P1	Oktett gemäß (N026.700)
	P2	Oktett gemäß (N026.800)
	Data	Optionaler Oktettstring gemäß 11.5.5
	LeFeld	Optionaler Oktettstring gemäß 11.5.6, welcher gemäß (N027.600) oder (N027.900) eine Zahl Ne enthält

	Kenc	Symmetrischer Schlüssel für die Verschlüsselung, (N029.900)d.2
	~~*SSCenc*~~	~~Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der Verschlüsselung verwendet wird, (N029.900)d.3~~
	Kmac	Symmetrischer Schlüssel für die MAC-Berechnung, (N029.900)d.4
	SSCmac	Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der MAC-Berechnung verwendet wird, (N029.900)d.5
Output:	CmdApdu	Gesicherte Kommando-APDU
Errors:	–	Keine

Zur Sicherung einer generischen Kommando-APDU sind folgende Schritte durchzuführen:

G2_N032.000 - (N032.000) K_externeWelt {K_Karte}
Wenn das optionale Datenfeld Data fehlt oder leer ist,
dann MUSS gelten: ProtectedData = '' (leerer Oktettstring).**[<=]**

G2_N032.100 - (N032.100) K_externeWelt {K_Karte}
Wenn das optionale Datenfeld Data vorhanden und nicht leer ist und im Klartext übertragen wird,
dann MUSS gelten:
a. Setze lenPDO = OctetLength(Data), falls lenPDO im Intervall
   1. [1, 127] liegt, gilt: LenP = I2OS(lenPDO, 1)
   2. [128, 255] liegt, gilt: LenP = '81' || I2OS(lenPDO, 1)
   3. [256, 65535] liegt, gilt: LenP = '82' || I2OS(lenPDO, 2)
b. Setze ProtectedData = '81 || LenP || Data'**[<=]**

G2_N032.190 - (N032.190) K_externeWelt {K_Karte}
SSCmac MUSS inkrementiert werden, das heißt: *SSCmac* = *SSCmac* + 1.**[<=]**

(N032.200) ist absichtlich leer.

G2_N032.200 - (N032.200) K_externeWelt {K_Karte}, Option_DES
~~Wenn das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird und *Kenc* ein 3TDES-Schlüssel ist, dann MUSS gelten:~~
~~a. *SSCenc* = *SSCenc* + 1~~
~~b. *C* = 3TDES_CBC_ENC(*Kenc*, *SSCenc*, PaddingIso(Data, 8))~~
~~c. Setze lenCDO = OctetLength(*C*) + 1, falls lenCDO im Intervall~~
~~1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)~~
~~2. [128, 255] liegt, gilt: LenC = '81' || I2OS(lenCDO, 1)~~
~~3. [256, 65535] liegt, gilt LenC = '82' || I2OS(lenCDO, 2)~~
~~d. Setze ProtectedData = '87 || LenC || 01 || *C*'~~**[<=]**

G2_N032.300 - (N032.300) K_externeWelt {K_Karte}
Wenn das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird und *Kenc* ein AES-Schlüssel ist, dann MUSS gelten:
a. IVenc = OS2I(AES_ENC(*Kenc*, I2OS(*SSCmac*, 16)))
b. C = AES_CBC_ENC(*Kenc*, IVenc, PaddingIso(Data, 16))
c. Setze lenCDO = OctetLength(*C*) + 1, falls lenCDO im Intervall

1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)
2. [128, 255] liegt, gilt: LenC = '81' || I2OS(lenCDO, 1)
3. [256, 65535] liegt, gilt LenC = '82' || I2OS(lenCDO, 2)

d. Setze ProtectedData = '87 || LenC || 01 || C '**[<=]**

G2_N032.400 - (N032.400) K_externeWelt {K_Karte}
Für LeDO MUSS gelten: Wenn das optionale LeFeld
a. fehlt, dann gilt: LeDO = '', (leerer Oktettstring).
b. vorhanden ist, dann gilt: LeDO = '97 || I2OS(OctetLength(LeFeld), 1) || LeFeld'**[<=]**

G2_N032.500 - (N032.500) K_externeWelt {K_Karte}
Wenn die Kanalnummer im CLA-Byte im Intervall
a. [0, 3] liegt, dann MUSS gesetzt werden CLA' = CLA OR '0C'
Hinweis CosH_1d4: Dadurch werden die Bits b4 und b3 in CLA gesetzt.
b. andernfalls MUSS gesetzt werden CLA' = CLA OR '20'
*Hinweis CosH_f56: Dadurch wird das Bit b6 in CLA gesetzt.***[<=]**

G2_N032.600 - (N032.600) K_externeWelt {K_Karte}
Wenn die Kanalnummer im CLA-Byte im Intervall
a. [0, 3] liegt, dann MUSS gelten *head* = CLA' || INS || P1 || P2
b. andernfalls MUSS gelten *head* = '89 04' || CLA' || INS || P1 || P2**[<=]**

(N032.700) Diese Anforderung ist absichtlich leer.

A_15862 - (N032.800) K_externeWelt {K_Karte}
Es MUSS gelten: tmpData = ProtectedData || LeDO**[<=]**

(N032.800)a ist absichtlich leer.

G2_N032.800.a - (N032.800)a K_externeWelt {K_Karte}, Option_DES
~~Wenn *Kmac* ein 3TDES-Schlüssel ist, dann MUSS gelten:~~
~~1. MACin = I2OS(*SSCmac*, 8)~~
~~2. Wenn die Kanalnummer im CLA-Byte größer gleich vier ist, oder OctetLength(tmpData) gleich null ist~~
~~i. dann gilt: MACin = MACin || head || tmpData~~
~~ii. sonst: MACin = MACin || PaddingIso(head, 8) || tmpData~~
~~3. MAC = CALCULATE_Retail_MAC(*Kmac*, MACin)~~**[<=]**

G2_N032.800.b - (N032.800)b K_externeWelt {K_Karte}
Wenn *Kmac* ein AES-Schlüssel ist, dann MUSS gelten:
1. MACin = I2OS(*SSCmac*, 16)
2. Wenn die Kanalnummer im CLA-Byte größer gleich vier ist, oder OctetLength(tmpData) gleich null ist
i. dann gilt: MACin = MACin || head || tmpData
ii. sonst: MACin = MACin || PaddingIso(head, 16) || tmpData
3. MAC = CalculateCMAC_IsoPadding(*Kmac*, MACin)**[<=]**

G2_N032.900 - (N032.900) K_externeWelt {K_Karte}
Für MDO MUSS gelten: MDO = '8E || I2OS(OctetLength(MAC), 1) || MAC'**[<=]**

Für newD MUSS gelten:

G2_N033.000.a - (N033.000)a K_externeWelt {K_Karte}
Wenn die Kanalnummer im CLA-Byte im Intervall [0, 3] liegt,
dann setze: newD = tmpData || MDO**[<=]**

G2_N033.000.b - (N033.000)b K_externeWelt {K_Karte}

Wenn die Kanalnummer im CLA-Byte größer als drei ist,
dann setze: newD = head || tmpData || MDO**[<=]**

G2_N033.100 - (N033.100) K_externeWelt {K_Karte}
Case 1: Wenn Data und LeFeld fehlen, dann MUSS gelten:
CmdApdu = Case4S(CLA', INS, P1, P2, newD, WildCardShort)**[<=]**

G2_N033.200 - (N033.200) K_externeWelt {K_Karte}
Case 2: Wenn Data fehlt und LeFeld vorhanden ist, dann MUSS gelten:
CmdApdu = Case4E(CLA', INS, P1, P2, newD, WildCardExtended)**[<=]**

G2_N033.300.a - (N033.300)a K_externeWelt {K_Karte}
Case 3S: Wenn Data vorhanden ist und LeFeld fehlt und OctetLength(newD) kleiner gleich 255 ist, dann MUSS gelten:
CmdApdu = Case4S(CLA', INS, P1, P2, newD, WildCardShort)**[<=]**

G2_N033.300.b - (N033.300)b K_externeWelt {K_Karte}
Case 3E: Wenn Data vorhanden ist und LeFeld fehlt und OctetLength(newD) größer als 255 ist, dann MUSS gelten:
CmdApdu = Case4E(CLA', INS, P1, P2, newD, WildCardExtended)**[<=]**

G2_N033.400 - (N033.400) K_externeWelt {K_Karte}
Case 4: Wenn Data und LeFeld vorhanden sind, dann MUSS gelten:
CmdApdu = Case4E(CLA', INS, P1, P2, newD, WildCardExtended)**[<=]**

Hinweis CosH_da9: Zu (N033.100) und (N033.300)a: Gemäß 11.7.1 (11.7.3) hat die korrespondierende Antwort-APDU zu einer Case 1 (Case 3) Kommando-APDU niemals Antwortdaten. Zudem werden in 13.3 ausschließlich symmetrische Verfahren verwendet. Daraus folgt, dass eine gesicherte Antwort-APDU zu einer ungesicherten Case 1 (Case 3) APDU niemals mehr als 256 Oktette Antwortdaten enthält. Deshalb wird hier für die gesicherte Case 1 (Case 3) Kommando-APDU eine Case 4 Short Kommando-APDU verwendet.

Hinweis CosH_fb6: Zu (N033.200) und (N033.400): Gemäß 11.7.2.1 (11.7.4.1) hat die korrespondierende Antwort-APDU zu einer Case 2 Short (Case 4 Short) Kommando-APDU bis zu 256 Oktette Antwortdaten. Gemäß 13.3 ist es deshalb möglich, dass eine gesicherte Antwort-APDU zu einer ungesicherten Case 2 Short (Case 4 Short) APDU mehr als 256 Oktette Antwortdaten enthält. Deshalb wird hier für gesicherte Case 2 Short (Case 4 Short) Kommando-APDU eine Case 4 Extended Kommando-APDU verwendet.

13.3 Sicherung einer Antwort-APDU

Dieses Unterkapitel beschreibt, wie eine Antwort-APDU (siehe 11.6) zu sichern ist. Die hier beschriebenen Regeln richten sich an das COS. Die Instanz, welche die korrespondierende Kommando-APDU gesendet hat und typischerweise diese gesicherte Antwort-APDU entgegennimmt, führt die entsprechenden Umkehroperationen durch, um in den Besitz der ungesicherten Antwort-APDU zu gelangen.

Generell wird hier ein Subset der Regeln aus [ISO/IEC 7816-4#10] beschrieben, wobei

- alle Antwort-APDU mit Integritätsschutz übertragen werden,
- der Trailer stets mit Integritätsschutz übertragen wird,
- Response-Daten als Klartext oder verschlüsselt übertragen werden.

Hier sei angemerkt, dass das COS anhand der verwendeten Zugriffsbedingung (Wert von *flagRspEnc*, siehe (N022.600)d) entscheidet, ob vorhandene Responsedaten im Klartext oder als Chiffrat übertragen werden.

Wie in 11.6 beschrieben, besteht eine Antwort-APDU generisch betrachtet aus dem optionalen Datenfeld Data und dem Trailer. Somit gelten folgende Definitionen:

Tabelle 94: CosT_4d6: Definition von Funktionsparametern zur Sicherung einer Antwort-APDU

Input:	Data	Optionaler Oktettstring gemäß 11.6.1
	Trailer	Oktettstring gemäß 11.6.2
	Kenc	Symmetrischer Schlüssel für die Verschlüsselung, (N029.900)d.2
	~~*SSCenc*~~	~~Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der Verschlüsselung verwendet wird, (N029.900)d.3~~
	Kmac	Symmetrischer Schlüssel für die MAC-Berechnung, (N029.900)d.4
	SSCmac	beliebige nicht-negative Zahl, die als Send Sequence Counter bei der MAC-Berechnung verwendet wird, (N029.900)d.5
Output:	RspApdu	Gesicherte Antwort-APDU
Errors:	–	Keine

Zur Sicherung einer generischen Antwort-APDU sind folgende Schritte durchzuführen:

G2_N033.500 - (N033.500) K_COS
Wenn das optionale Datenfeld Data fehlt oder leer ist, dann MUSS gelten:
ProtectedData = '' (leerer Oktettstring).**[<=]**

G2_N033.600 - (N033.600) K_COS
Wen das optionale Datenfeld Data vorhanden und nicht leer ist und im Klartext übertragen wird, dann MUSS gelten:
a. Setze lenPDO = OctetLength(Data), falls lenPDO im Intervall
   1. [1, 127] liegt, gilt: LenP = I2OS(lenPDO, 1)
   2. [128, 255] liegt, gilt: LenP = '81' || I2OS(lenPDO, 1)
   3. [256, 65535] liegt, gilt: LenP = '82' || I2OS(lenPDO, 2)
b. Setze ProtectedData = '81 || LenP || Data'**[<=]**

G2_N033.690 - (N033.690) K_COS
SSCmac MUSS inkrementiert werden, das heißt: *SSCmac* = *SSCmac* + 1.**[<=]**

(N033.700) ist absichtlich leer.

G2_N033.700 - (N033.700) K_COS, Option_DES
~~Wenn das optionale Datenfeld Data vorhanden und nicht leer ist und verschlüsselt übertragen wird (*flagRspEnc* ist True, siehe (N029.900)d.7 und (N022.600)d) und *Kenc* ein 3TDES-Schlüssel ist, dann MUSS gelten:~~
~~a. *SSCenc* = *SSCenc* + 1~~
~~b. *C* = 3TDES_CBC_ENC(*Kenc*, *SSCenc*, PaddingIso(Data, 8))~~
~~c. Setze lenCDO = OctetLength(*C*) + 1, falls lenCDO im Intervall~~

~~1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)~~
~~2. [128, 255] liegt, gilt: LenC = '81' || I2OS(lenCDO, 1)~~
~~3. [256, 65535] liegt, gilt LenC = '82' || I2OS(lenCDO, 2)~~
~~d. Setze ProtectedData = '87 || LenC || 01 || C'~~**[<=]**

G2_N033.800 - (N033.800) K_COS
Wenn das optionale Datenfeld Data vorhanden und nicht leer ist und verschlüsselt übertragen wird (*flagRspEnc* ist True, siehe (N029.900)d.7 und (N022.600)d) und *Kenc* ein AES-Schlüssel ist, dann MUSS gelten:
a. IVenc = OS2I(AES_ENC(*Kenc*, I2OS(*SSCmac*, 16)))
b. C = AES_CBC_ENC(*Kenc*, IVenc, PaddingIso(Data, 16))
c. Setze lenCDO = OctetLength(*C*) + 1, falls lenCDO im Intervall
1. [1, 127] liegt, gilt: LenC = I2OS(lenCDO, 1)
2. [128, 255] liegt, gilt: LenC = '81' || I2OS(lenCDO, 1)
3. [256, 65535] liegt, gilt LenC = '82' || I2OS(lenCDO, 2)
d. Setze ProtectedData = '87 || LenC || 01 || C '**[<=]**

G2_N033.900 - (N033.900) K_COS
Für TDO MUSS gelten: TDO = '99 02 || Trailer'**[<=]**

(N034.000) Diese Anforderung ist absichtlich leer.

(N034.100)a ist absichtlich leer.

G2_N034.100.a - (N034.100)a K_COS, Option_DES
~~Wenn *Kmac* ein 3TDES-Schlüssel ist, dann MUSS gelten:~~
~~1. MACin = I2OS(*SSCmac*, 8)~~
~~2. MACin = MACin || ProtectedData~~
~~3. MACin = MACin || TDO~~
~~4. MAC = CALCULATE_Retail_MAC(*Kmac*, MACin)~~**[<=]**

G2_N034.100.b - (N034.100)b K_COS
Wenn *Kmac* ein AES-Schlüssel ist, dann MUSS gelten:
1. MACin = I2OS(*SSCmac*, 16) || ProtectedData || TDO
2. MAC = CalculateCMAC_IsoPadding(*Kmac*, MACin)**[<=]**

G2_N034.200 - (N034.200) K_COS
Für MDO MUSS gelten: MDO = '8E || I2OS(OctetLength(MAC), 1) || MAC'**[<=]**

G2_N034.300 - (N034.300) K_COS
Für die gesicherte Antwort-APDU RspApdu MUSS gelten:
a. RspApdu.Datenfeld = ProtectedData || TDO || MDO
b. RspApdu.Trailer = Trailer**[<=]**

14 Kommandos (normativ)

Tabelle 95: CosT_c84: Kommandos alphabetisch und numerisch sortiert

Kommando alphabetisch	CLA INS	siehe	Kommandos numerisch	CLA INS	siehe
ACITVATE	'00 44'	14.2.1	DEACTIVATE	'00 04'	14.2.3
ACTIVATE RECORD	'00 08'	14.4.1	DEACTIVATE RECORD	'00 06'	14.4.3
APPEND RECORD	'00 E2'	14.4.2	ACTIVATE RECORD	'00 08'	14.4.1
CHANGE REFERENCE DATA	'00 24'	14.6.1	ERASE RECORD	'00 0C'	14.4.5
CREATE	'00 E0'	14.2.2	ERASE BINARY	'00 0E'	14.3.1
DEACTIVATE	'00 04'	14.2.3	VERIFY	'00 20'	14.6.6
DEACTIVATE RECORD	'00 06'	14.4.3	MANAGE SECURITY ENVIRONMENT	'00 22'	14.9.9
DELETE	'00 E4'	14.2.4	CHANGE REFERENCE DATA	'00 24'	14.6.1
DELETE RECORD	'80 0C'	14.4.4	DISABLE VERIFICATION REQUIREMENT	'00 26'	14.6.2
DISABLE VERIFICATION REQUIREMENT	'00 26'	14.6.2	ENABLE VERIFICATION REQUIREMENT	'00 28'	14.6.3
ENABLE VERIFICATION REQUIREMENT	'00 28'	14.6.3	PERFORM SECURITY OPERATION	'00 2A'	14.8
ENVELOPE	'00 C2'	14.9.1	RESET RETRY COUNTER	'00 2C'	14.6.5
ERASE BINARY	'00 0E'	14.3.1	ACITVATE	'00 44'	14.2.1
ERASE RECORD	'00 0C'	14.4.5	GENERATE ASYMMETRIC KEY PAIR	'00 46'	14.9.3
EXTERNAL AUTHENTICATE	'00 82'	14.7.1	MANAGE CHANNEL	'00 70'	14.9.8

FINGERPRINT	'80 FA'	14.9.2		EXTERNAL AUTHENTICATE	'00 82'	14.7.1
GENERAL AUTHENTICATE	'00 86'	14.7.2		MUTUAL AUTHENTICATE	'00 82'	14.7.1
GENERATE ASYMMETRIC KEY PAIR	'00 46'	14.9.3		GET CHALLENGE	'00 84'	14.9.4
GET CHALLENGE	'00 84'	14.9.4		GENERAL AUTHENTICATE	'00 86'	14.7.2
GET DATA	'00 CA'	14.5.1		INTERNAL AUTHENTICATE	'00 88'	14.7.4
GET PIN STATUS	'80 20'	14.6.4		SEARCH BINARY	'00 A0'	14.3.3
GET RANDOM	'80 84'	14.9.5		SEARCH RECORD	'00 A2'	14.4.7
GET RESPONSE	'00 C0'	14.9.6		SELECT	'00 A4'	14.2.6
GET SECURITY STATUS KEY	'80 82'	14.7.3		READ BINARY	'00 B0'	14.3.2
INTERNAL AUTHENTICATE	'00 88'	14.7.4		READ RECORD	'00 B2'	14.4.6
LIST PUBLIC KEY	'80 CA'	14.9.7		GET RESPONSE	'00 C0'	14.9.6
LOAD APPLICATION	'00 EA'	14.2.5		ENVELOPE	'00 C2'	14.9.1
MANAGE CHANNEL	'00 70'	14.9.8		GET DATA	'00 CA'	14.5.1
MANAGE SECURITY ENVIRONMENT	'00 22'	14.9.9		WRITE BINARY	'00 D0'	14.3.6
MUTUAL AUTHENTICATE	'00 82'	14.7.1		WRITE RECORD	'00 D2'	14.4.9
PERFORM SECURITY OPERATION	'00 2A'	14.8		UPDATE BINARY	'00 D6'	14.3.5
PUT DATA	'00 DA'	14.5.2		PUT DATA	'00 DA'	14.5.2
READ BINARY	'00 B0'	14.3.2		UPDATE RECORD	'00 DC'	14.4.8
READ RECORD	'00 B2'	14.4.6		CREATE	'00 E0'	14.2.2

RESET RETRY COUNTER	'00 2C'	14.6.5		APPEND RECORD	'00 E2'	14.4.2
SEARCH BINARY	'00 A0'	14.3.3		DELETE	'00 E4'	14.2.4
SEARCH RECORD	'00 A2'	14.4.7		TERMINATE DF	'00 E6'	14.2.8
SELECT	'00 A4'	14.2.6		TERMINATE	'00 E8'	14.2.9
SET LOGICAL EOF	'80 0E'	14.3.4		LOAD APPLICATION	'00 EA'	14.2.5
TERMINATE	'00 E8'	14.2.9		TERMINATE CARD USAGE	'00 FE'	14.2.7
TERMINATE CARD USAGE	'00 FE'	14.2.7		DELETE RECORD	'80 0C'	14.4.4
TERMINATE DF	'00 E6'	14.2.8		SET LOGICAL EOF	'80 0E'	14.3.4
UPDATE BINARY	'00 D6'	14.3.5		GET PIN STATUS	'80 20'	14.6.4
UPDATE RECORD	'00 DC'	14.4.8		GET SECURITY STATUS KEY	'80 82'	14.7.3
VERIFY	'00 20'	14.6.6		GET RANDOM	'80 84'	14.9.5
WRITE BINARY	'00 D0'	14.3.6		LIST PUBLIC KEY	'80 CA'	14.9.7
WRITE RECORD	'00 D2'	14.4.9		FINGERPRINT	'80 FA'	14.9.2

A_15865 - (N034.400) K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS Kommando-APDU unterstützt, die in Kapitel 14 nicht aufgeführt sind.**[<=]**

14.1 Roll-Verhalten

In den folgenden Unterkapiteln ist gelegentlich die Rede davon, dass der persistente Speicher mittels "Roll-Forward" oder "Roll-Back" zu verändern ist. Als Oberbegriff wird "Transaktionsschutz" verwendet, um auszudrücken, dass Roll-Forward oder Roll-Back gemeint ist. Kurz gesagt verbirgt sich dahinter Folgendes: Das persistente Speichern von Informationen dauert aus technischen Gründen einige Millisekunden. Da Smartcards aus technischen Gründen nicht in der Lage sind, einen Ausfall der Spannungsversorgung zu puffern, ist es denkbar, dass der Ausfall zu einem Zeitpunkt geschieht, in welchem der Zustand des persistenten Speichers in einem undefinierten Zustand ist. Der

Transaktionsschutz legt dann fest, wie mit diesem möglicherweise undefiniertem Zustand umzugehen ist.

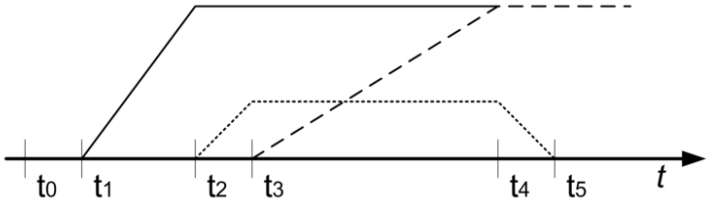


Abbildung 2: CosA_8fc: Zeitlicher Ablauf eines Roll-Back-Kommandos

Für die Bearbeitung eines Kommandos mit Transaktionsschutz werden, wie in CosA_8fc gezeigt, die folgenden Zeitpunkte definiert:

- Zum Zeitpunkt t_0 werde das erste Bit der Kommando-APDU über die physikalische Schnittstelle (siehe CosA_e09) zur Smartcard gesendet.
- Zum Zeitpunkt t_1 sei die Kommandobearbeitung so weit vorgeschritten, dass der Zwischenspeicher bereit ist, befüllt zu werden.
- Zum Zeitpunkt t_2 sei der Zwischenspeicher komplett befüllt, aber sein Inhalt noch nicht als gültig gekennzeichnet.
- Zum Zeitpunkt t_3 sei der Inhalt des Zwischenspeichers als gültig gekennzeichnet und die eigentliche, persistente Änderung werde gestartet.
- Zum Zeitpunkt t_4 sei die eigentliche, persistente Änderung abgeschlossen, aber der Inhalt des Zwischenpuffers sei noch als gültig gekennzeichnet.
- Zum Zeitpunkt t_5 sei der Inhalt des Zwischenspeichers als ungültig gekennzeichnet.
- Zum Zeitpunkt t_e (in CosA_8fc nicht dargestellt) habe die Smartcard das letzte Bit der Antwort-APDU über die physikalische Schnittstelle versendet.

A_15915 - (N034.500) K_TST
Dieses Dokument legt nicht fest, in welchem zeitlichen Zusammenhang t_e zu den anderen Zeiten aus der Menge $\{t_1, t_2, t_3, t_4, t_5\}$ steht. ~~Damit KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass t_e zu einem beliebigen Zeitpunktes nach t_0 erfolgt.**[<=]**

14.1.1 Roll-Back

Roll-Back legt fest, dass die durch den Spannungsausfall unterbrochene Aktion rückgängig zu machen ist, wenn wieder eine Versorgungsspannung anliegt. Typischerweise wird dazu vor Durchführung der Änderung der *ursprüngliche* Inhalt in einen Zwischenspeicher geschrieben, dann die Änderung durchgeführt und anschließend der Inhalt des Zwischenspeichers als ungültig gekennzeichnet.

A_15916 - (N034.600) K_COS
Findet der Ausfall der Versorgungsspannung

a. vor t_4 statt, so ist entweder noch keine Änderung erfolgt, oder der Inhalt des Zwischenpuffers definitiv auf gültig gesetzt und der (hier ursprüngliche) Inhalt des Zwischenspeichers MUSS nach Wiederanlegen der Versorgungsspannung wiederhergestellt werden.

b. nach t_5 statt, so ist der Inhalt des Zwischenpuffers definitiv auf ungültig gesetzt und damit eine Wiederherstellung des ursprünglichen Zustandes unmöglich.

c. zwischen t_4 und t_5 statt, so hängt es vom Zufall ab, ob der Zustand des Zwischenspeichers als gültig oder ungültig beurteilt wird (physikalische Speicher haben mitnichten ein zeit- oder wertediskretes Verhalten).
In diesem Fall MUSS entweder der ursprüngliche Zustand rekonstruiert oder der neue Zustand beibehalten werden.

[<=]

14.1.2 Roll-Forward

Roll-Forward legt fest, dass die durch den Spannungsausfall unterbrochene Aktion fortzusetzen ist, wenn wieder eine Versorgungsspannung anliegt. Typischerweise wird dazu vor Durchführung der Änderung der *neue* Inhalt in einen Zwischenspeicher geschrieben, dann die Änderung durchgeführt und anschließend der Inhalt des Zwischenspeichers als ungültig gekennzeichnet.

A_15917 - (N034.700) K_COS
Findet der Ausfall der Versorgungsspannung

d. vor t_2 statt, so ist der Inhalt des Zwischenspeichers definitiv nicht auf gültig gesetzt, und damit ist ein Wechsel zum neuen Zustand unmöglich.

e. nach t_3 statt, so ist der Inhalt des Zwischenpuffers definitiv auf gültig gesetzt, und der (hier neue) Inhalt des Zwischenspeichers MUSS nach Wiederanlegen der Versorgungsspannung wiederhergestellt werden.

f. zwischen t_2 und t_3 statt, so hängt es vom Zufall ab, ob der Zustand des Zwischenspeichers als gültig oder ungültig beurteilt wird (physikalische Speicher haben mitnichten zeit- oder wertediskretes Verhalten).
In diesem Fall MUSS entweder der ursprüngliche Zustand beibehalten oder der neue Zustand gesetzt werden.

[<=]

14.2 Management des Objektsystems

14.2.1 ACTIVATE

Das Kommando ACTIVATE aktiviert reversibel ein Objekt. Ein betroffenes File wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses ACTIVATE-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*). Wenn ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, dann wird dieses durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachricht enthalten ist.

14.2.1.1 Use Case Aktivieren eines Ordners oder einer Datei

In dieser Variante wird ein Ordner oder eine Datei aktiviert und die APDU des ACTIVATE-Kommandos enthält einen Parameter:

A_15918 - (N034.798) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass das aktuelle File zu aktivieren ist.**[<=]**

A_15919 - (N034.800) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_2b6 verwendet werden.

Tabelle 96: CosT_2b6: ACTIVATE aktuelles File

	Inhalt	Beschreibung
CLA	'00'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'44'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	*mode*, der Wert '00' zeigt an, dass das aktuelle File zu aktivieren ist
P2	'00'	–

[<=]

14.2.1.2 Use Case Aktivieren eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt aktiviert und die APDU des ACTIVATE-Kommandos enthält zwei Parameter:

A_15920 - (N034.810) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein privates oder symmetrisches Schlüsselobjekt zu aktivieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.**[<=]**

A_15921 - (N034.812) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

A_15922 - (N034.814) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_079 verwendet werden.

Tabelle 97: CosT_079: *ACTIVATE* privates oder symmetrisches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'44'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'20'	*mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	*reference* auf ein Schlüsselobjekt

[<=]

14.2.1.3 Use Case Aktivieren eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt aktiviert und die APDU des ACTIVATE-Kommandos enthält zwei Parameter:

A_15923 - (N034.820) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein Schlüsselobjekt zu aktivieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.**[<=]**

A_15924 - (N034.822) K_externeWelt {K_Karte}
Der Parameter *reference* MUSS eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt enthalten.**[<=]**

A_15925 - (N034.824) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_ded verwendet werden.

Tabelle 98: CosT_ded: *ACTIVATE* öffentliches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'44'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'21'	*mode*, hier: Schlüsselreferenz im Datenteil
P2	'00'	-
Data	'XX…XX'	'83 – I2OS(OctetLength(*reference*), 1) – *reference*'

[<=]

14.2.1.4 Use Case Aktivieren eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt aktiviert und die APDU des ACTIVATE-Kommandos enthält zwei Parameter:

A_15926 - (N034.830) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein Passwortobjekt zu aktivieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.**[<=]**

A_15927 - (N034.832) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.**[<=]**

A_15928 - (N034.834) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_ee1 verwendet werden.

Tabelle 99: CosT_ee1: *ACTIVATE* Passwortobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'44'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'10'	*mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	'XX'	*reference* auf ein Passwortobjekt

[<=]

14.2.1.5 Antwort der Karte auf Aktivieren eines Files

Tabelle 100: CosT_b59: *ACTIVATE* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Aktivierung

Tabelle 101: CosT_60b: *ACTIVATE* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	ObjectTerminated	Objekt befindet sich im Zustand "Termination state"
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	VolatileKeyWithoutLCS	volatile Schlüssel vom Kommando nicht unterstützt
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'6A 88'	KeyNotFound	Schlüsselobjekt nicht gefunden
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_60e: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_15929 - (N034.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das ACTIVATE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.2.1.6 Kommandoabarbeitung innerhalb der Karte

G2_N035.000.a - (N035.000)a K_COS
Das COS MUSS die ACTIVATE-Varianten aus
- 14.2.1.1- Use Case Aktivieren eines Ordners oder einer Datei,
- 14.2.1.2- Use Case Aktivieren eines privaten oder symmetrischen Schlüsselobjektes,
- 14.2.1.3- Use Case Aktivieren eines öffentlichen Schlüsselobjektes und
- 14.2.1.4- Use Case Aktivieren eines Passwortobjektes

unterstützen.**[<=]**

A_15930 - (N035.000)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere ACTIVATE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N035.100.a.1 - (N035.100)a.1 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und *channelContext. currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N035.100.a.2 - (N035.100)a.2 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und *channelContext. currentEF* (siehe (N029.900)m) nicht auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentFolder* gesetzt werden.**[<=]**

G2_N035.100.b - (N035.100)b K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '20' oder '21' besitzt, dann gilt
affectedObject = SearchKey(
channelContext. currentFolder,
reference,
"WildCard"
). Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N035.100.c - (N035.100)c K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '10' besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N035.110 - (N035.110) K_COS
Wenn *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer VolatileKeyWithoutLCS terminieren.**[<=]**

G2_N035.200 - (N035.200) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N035.300.a - (N035.300)a K_COS
Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert "Termination state" besitzt, genau dann MUSS das Kommando mit dem Trailer ObjectTerminated terminieren.**[<=]**

G2_N035.300.b - (N035.300)b K_COS
Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert "Operational state (active)" besitzt, dann MUSS als Trailer NoError verwendet werden.**[<=]**

G2_N035.400 - (N035.400) K_COS
Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert "Operational state (active)" gesetzt werden.**[<=]**

A_15931 - (N035.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_15932 - (N035.600) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N035.700 - (N035.700) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_15933 - (N035.800) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_60b ist herstellerspezifisch.

b. Jeder Trailer in CosT_60b MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.2.2 CREATE

A_15934 - (N035.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das CREATE-Kommando gemäß [ISO/IEC 7816-9]
a. unterstützt oder
b. ablehnt.**[<=]**

Hinweis CosH_f64: Die entsprechende Funktionalität dieses Kommandos wird im Rahmen dieses Dokumentes durch das Kommando LOAD APPLICATION *(siehe 14.2.5) bereitgestellt.*

14.2.3 DEACTIVATE

Das Kommando DEACTIVATE deaktiviert reversibel ein Objekt. Ein betroffenes File wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses DEACTIVATE-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*). Wenn ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, dann wird dieses durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachricht enthalten ist.

14.2.3.1 Use Case Deaktivieren eines Ordners oder einer Datei

In dieser Variante wird ein Ordner oder eine Datei deaktiviert und die APDU des DEACTIVATE-Kommandos enthält einen Parameter:

A_15936 - (N035.998) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass das aktuelle File zu deaktivieren ist.**[<=]**

A_15937 - (N036.000) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_e37 verwendet werden.

Tabelle 102: CosT_e37: *DEACTIVATE* aktuelles File

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'04'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	*mode*, der Wert '00' zeigt an, dass das aktuelle File zu deaktivieren ist
P2	'00'	–

[<=]

14.2.3.2 Use Case Deaktivieren eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt deaktiviert und die APDU des DEACTIVATE-Kommandos enthält zwei Parameter:

A_15938 - (N036.010) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein privates oder symmetrisches Schlüsselobjekt zu deaktivieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.**[<=]**

A_15939 - (N036.012) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

A_15940 - (N036.014) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_a2f verwendet werden.

Tabelle 103: CosT_a2f: *DEACTIVATE* privates oder symmetrisches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'04'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'20'	*mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	*reference* auf ein Schlüsselobjekt

[<=]

14.2.3.3 Use Case Deaktivieren eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt deaktiviert und die APDU des DEACTIVATE-Kommandos enthält zwei Parameter:

A_15941 - (N036.020) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein Schlüsselobjekt zu deaktivieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.**[<=]**

A_15942 - (N036.022) K_externeWelt {K_Karte}
Der Parameter *reference* MUSS eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt enthalten.**[<=]**

A_15943 - (N036.024) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_ac5 verwendet werden.

Tabelle 104: CosT_ac5: *DEACTIVATE* öffentliches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'04'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'21'	*mode*, hier: Schlüsselreferenz im Datenteil
P2	'00'	-
Data	'XX…XX'	'83 – I2OS(OctetLength(*reference*), 1) – *reference*'

[<=]

14.2.3.4 Use Case Deaktivieren eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt deaktiviert und die APDU des DEACTIVATE-Kommandos enthält zwei Parameter:

A_15944 - (N036.030) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein Passwortobjekt zu deaktivieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.**[<=]**

A_15945 - (N036.032) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.**[<=]**

A_15946 - (N036.034) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_e19 verwendet werden.

Tabelle 105: CosT_e19: *DEACTIVATE* Passwortobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'04'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'10'	*mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	'XX'	*reference* auf ein Passwortobjekt

[<=]

14.2.3.5 Antwort der Karte auf Deaktivieren eines Files

Tabelle 106: CosT_294: *DEACTIVATE* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Deaktivierung

Tabelle 107: CosT_93a: *DEACTIVATE* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	ObjectTerminated	Objekt befindet sich im Zustand "Termination state"
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	VolatileKeyWithoutLCS	volatile Schlüssel vom Kommando nicht unterstützt
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'6A 88'	KeyNotFound	Schlüsselobjekt nicht gefunden
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_94c: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_15947 - (N036.100) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das DEACTIVATE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.2.3.6 Kommandoabarbeitung innerhalb der Karte

G2_N036.200.a - (N036.200)a K_COS
Das COS MUSS die DEACTIVATE-Varianten aus
- 14.2.3.1- Use Case Deaktivieren eines Ordners oder einer Datei,
- 14.2.3.2- Use Case Deaktivieren eines privaten oder symmetrischen Schlüsselobjektes,
- 14.2.3.3- Use Case Deaktivieren eines öffentlichen Schlüsselobjektes und
- 14.2.3.4- Use Case Deaktivieren eines Passwortobjektes

unterstützen.**[<=]**

A_15948 - (N036.200)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere DEACTIVATE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N036.300.a.1 - (N036.300)a.1 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und*channelContext*. *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N036.300.a.2 - (N036.300)a.2 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und*channelContext*. *currentEF* (siehe (N029.900)m) nicht auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentFolder* gesetzt werden.**[<=]**

G2_N036.300.b - (N036.300)b K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '20' oder '21' besitzt, dann gilt
affectedObject = SearchKey(
channelContext. *currentFolder*,
reference,
"WildCard"
). Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N036.300.c - (N036.300)c K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '10' besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N036.310 - (N036.310) K_COS
Wenn *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer VolatileKeyWithoutLCS terminieren.**[<=]**

G2_N036.400 - (N036.400) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N036.500.a - (N036.500)a K_COS
Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert "Termination state" besitzt, genau dann MUSS das Kommando mit dem Trailer ObjectTerminated terminieren.**[<=]**

G2_N036.500.b - (N036.500)b K_COS
Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert "Operational state (deactived)" besitzt, dann MUSS als Trailer NoError verwendet werden.**[<=]**

G2_N036.600 - (N036.600) K_COS
Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert "Operational state (deactivated)" gesetzt werden.**[<=]**

A_15949 - (N036.700) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_15950 - (N036.800) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N036.900 - (N036.900) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_15951 - (N037.000) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_93a ist herstellerspezifisch.
b. Jeder Trailer in CosT_93a MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.2.4 DELETE

Das Kommando DELETE löscht Objekte aus dem Objektsystem. Ein zu löschendes File wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses DELETE-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*). Welches Schlüsselobjekt oder welches Passwortobjekt vom Kommando betroffen ist, wird durch eine Schlüssel- oder Passwortreferenz bestimmt, die

in der Kommandonachricht enthalten ist. Für dieses Kommando gelten Restriktionen, siehe (N099.500).

14.2.4.1 Use Case Löschen eines Ordners oder einer Datei

In dieser Variante wird ein Ordner oder eine Datei gelöscht und die APDU des DELETE-Kommandos enthält einen Parameter:

A_15952 - (N037.098) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass das aktuelle File zu löschen ist.**[<=]**

A_15953 - (N037.100) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_799 verwendet werden.

Tabelle 108: CosT_799: *DELETE aktuelles File*

	Inhalt	Beschreibung
CLA	'00'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'E4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	*mode*, der Wert '00' zeigt an, dass das aktuelle File zu löschen ist
P2	'00'	–

[<=]

14.2.4.2 Use Case Löschen eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt gelöscht und die APDU des DELETE-Kommandos enthält zwei Parameter:

A_15954 - (N037.110) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein privates oder symmetrisches Schlüsselobjekt zu löschen ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.**[<=]**

A_15955 - (N037.112) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

A_15956 - (N037.114) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_57c verwendet werden.

Tabelle 109: CosT_57c: *DELETE* privates oder symmetrisches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'20'	*mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	*reference* auf ein Schlüsselobjekt

[<=]

14.2.4.3 Use Case Löschen eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt gelöscht und die APDU des DELETE-Kommandos enthält zwei Parameter:

A_15957 - (N037.120) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein Schlüsselobjekt zu löschen ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.**[<=]**

A_15958 - (N037.122) K_externeWelt {K_Karte}
Der Parameter *reference* MUSS eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt enthalten.**[<=]**

A_15959 - (N037.124) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_57b verwendet werden.

Tabelle 110: CosT_57b: *DELETE* öffentliches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'21'	*mode*, hier: Schlüsselreferenz im Datenteil
P2	'00'	-
Data	'XX…XX'	'83 – I2OS(OctetLength(*reference*), 1) – *reference*'

[<=]

14.2.4.4 Use Case Löschen eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt gelöscht und die APDU des DELETE-Kommandos enthält zwei Parameter:

A_15960 - (N037.130) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass ein Passwortobjekt zu löschen ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.**[<=]**

A_15961 - (N037.132) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.**[<=]**

A_15962 - (N037.134) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_4e7 verwendet werden.

Tabelle 111: CosT_4e7: *DELETE* Passwortobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'10'	*mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	'XX'	*reference* auf ein Passwortobjekt

[<=]

14.2.4.5 Antwort der Karte auf Löschen eines Files

Tabelle 112: CosT_ee2: *DELETE* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Löschoperation

Tabelle 113: CosT_6c1: *DELETE* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'6A 88'	KeyNotFound	Schlüsselobjekt nicht gefunden
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_14b: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_15963 - (N037.200) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das DELETE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.2.4.6 Kommandoabarbeitung innerhalb der Karte

G2_N037.300.a - (N037.300)a K_COS
Das COS MUSS die DELETE-Varianten aus
- 14.2.4.1- Use Case Löschen eines Ordners oder einer Datei,
- 14.2.4.2- Use Case Löschen eines privaten oder symmetrischen Schlüsselobjektes,
- 14.2.4.3- Use Case Löschen eines öffentlichen Schlüsselobjektes und

- 14.2.4.4- Use Case Löschen eines Passwortobjektes
unterstützen.**[<=]**

A_15964 - (N037.300)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere DELETE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N037.400.a.1 - (N037.400)a.1 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und*channelContext. currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N037.400.a.2 - (N037.400)a.2 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und*channelContext. currentEF* (siehe (N029.900)m) nicht auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentFolder* gesetzt werden.**[<=]**

G2_N037.400.b - (N037.400)b K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '20' oder '21' besitzt, dann gilt
affectedObject = SearchKey(
channelContext. currentFolder,
reference,
"WildCard"
). Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N037.400.c - (N037.400)c K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '10' besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N037.500 - (N037.500) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N037.600 - (N037.600) K_COS
Die Löschoperation MUSS mit Transaktionsschutz durchgeführt werden.**[<=]**

G2_N037.690 - (N037.690) K_COS
Wenn die Kommandos-APDU dieses DELETE-Kommandos gemäß 13.2 mittels Sessionkeys gesichert war, dann MUSS die korrespondierende Antwort-APDU gemäß 13.3 mit denselben Sessionkeys gesichert sein, auch wenn als Seiteneffekt von (N037.700) Sessionkeys gelöscht werden.**[<=]**

A_15965 - (N037.700)a K_COS
Wenn *affectedObject* vom Typ Ordner ist, dann MÜSSEN die folgenden Schritte ausgeführt werden:**[<=]**

G2_N037.700.a.1 - (N037.700)a.1 K_COS
Schritt 1:clearSecurityStatusFolder(*affectedObject*) MUSS ausgeführt werden.**[<=]**

G2_N037.700.a.2 - (N037.700)a.2 K_COS
Aus *dfSpecificPasswordList* MÜSSEN alle Einträge entfernt werden, die zu *affectedObject* gehören.**[<=]**

G2_N037.700.a.3 - (N037.700)a.3 K_COS
In *keyReferenceList* MÜSSEN alle Einträge auf den Wert "leer" gesetzt werden, die auf Schlüsselobjekte verweisen, die *affectedObject* zugeordnet sind.**[<=]**

G2_N037.700.a.4 - (N037.700)a.4 K_COS
currentEF MUSS auf den Wert "undefined" gesetzt werden.**[<=]**

G2_N037.700.a.5 - (N037.700)a.5 K_COS
currentFolder MUSS auf den Vater von *affectedObject* gesetzt werden.**[<=]**

G2_N037.700.a.6 - (N037.700)a.6 K_COS
Aus *allPublicKeyList* MÜSSEN alle Einträge entfernt werden, die *affectedObject* oder einem seiner Unterordner zugeordnet sind.**[<=]**

G2_N037.700.a.7 - (N037.700)a.7 K_COS
affectedObject MUSS inklusive Subtree (also rekursiv aller seiner Kinder) gelöscht werden.**[<=]**

G2_N037.700.b - (N037.700)b K_COS
Wenn *affectedObject* vom Typ Datei ist, dann MUSS
1. *currentEF* auf den Wert "undefined" gesetzt werden und
2. *affectedObject* gelöscht werden.**[<=]**

G2_N037.700.c - (N037.700)c K_COS
Wenn *affectedObject* vom Typ Symmetrisches Authentisierungsobjekt oder symmetrisches Kartenverbindungsobjekt ist, dann MUSS
1. clearSecurityStatusKey(*affectedObject*) ausgeführt werden und
2. *affectedObject* gelöscht werden.**[<=]**

G2_N037.700.d - (N037.700)d K_COS
Wenn *affectedObject* vom Typ Passwortobjekt ist, dann MUSS
1. clearPasswordStatus(*affectedObject*) ausgeführt werden und
2. *affectedObject* gelöscht werden.**[<=]**

G2_N037.700.e - (N037.700)e K_COS
Wenn *affectedObject* vom einen anderen Typ besitzt, als die zuvor genannten, dann MUSS *affectedObject* gelöscht werden.**[<=]**

G2_N037.800.a - (N037.800)a K_COS
Wenn *affectedObject* vom Typ Ordner oder Datei ist, dann MUSS vormals von diesem Objekt allokierter Speicher so freigegeben werden, dass er zum Anlegen anderer Objekte verwendbar ist.**[<=]**

A_15966 - (N037.800)b.1 K_COS
Wenn *affectedObject* ~~vom Typ Symmetrisches Authentisierungsobjekt oder Passwortobjekt~~ nicht vom Typ Ordner oder Datei ist, dann SOLL vormals von diesem Objekt allokierter Speicher so freigegeben werden, dass er zum Anlegen anderer Objekte verwendbar ist.**[<=]**

A_15967 - (N037.800)b.2 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS beim Löschen von Objekten, die weder vom Typ Ordner noch vom Typ Datei sind, allokierten Speicher nicht freigibt.**[<=]**

A_15968 - (N037.850) K_TST
Wenn *affectedObject* ein Schlüssel ist, der in einem Element von *keyReferenceList* eingetragen ist, dann ~~KANN~~ MUSS es für die funktionale Eignung zulässig sein, dass der zum gelöschten Schlüssel gehörende Eintrag in *keyReferenceList*
a. gelöscht wird oder
b. bestehen bleibt.**[<=]**

A_15969 - (N037.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_15970 - (N038.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N038.100 - (N038.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_15971 - (N038.200) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_6c1 ist herstellerspezifisch.
b. Jeder Trailer in CosT_6c1 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

G2_N038.300 - (N038.300) K_COS
Im Fehlerfall MÜSSEN *currentFolder* und *currentEF* unverändert auf dem Wert vor Ausführung des Kommandos belassen werden.**[<=]**

14.2.5 LOAD APPLICATION

Das Kommando LOAD APPLICATION wird verwendet, um neue Files im Objektsystem anzulegen. So ist es möglich,

- einen neuen Ordner inklusive Subtree,
- eine neue Datei inklusive Inhalt (transparent oder strukturiert)

anzulegen. Das neu angelegte File wird im *currentFolder* gespeichert. Typischerweise ist die beim Anlegen neuer Files zur Karte transferierte Datenmenge so groß, dass sie nicht in einer einzigen Kommando-APDU übertragbar ist. Deshalb unterstützt dieses Kommando "Command Chaining", siehe11.8.

14.2.5.1 Use Case Anlegen neues Objekt, nicht Ende der Kommandokette

Diese Variante ist zu wählen, wenn die Datenmenge nicht in einer Kommando-APDU übertragbar ist. Sie kommt zum Einsatz von der ersten bis zur vorletzten Kommando-APDU. Die letzte Kommando-APDU wird gemäß (N038.800) übertragen. In der hier beschriebenen Variante enthält die APDU des LOAD APPLICATION-Kommandos zwei Parameter.

A_15973 - (N038.400) K_externeWelt {K_Karte}
Das CLA-Byte MUSS anzeigen, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist. Dies wird durch den Wert CLA = '10' codiert.**[<=]**

A_15974 - (N038.500) K_externeWelt {K_Karte}
Der Parameter *cmdData* enthält Daten, welche das neu anzulegende File beschreiben. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem, herstellerspezifischem Inhalt. Die Länge von *cmdData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.**[<=]**

A_15975 - (N038.600) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_7b8 verwendet werden.

Tabelle 114: CosT_7b8: *LOAD APPLICATION* mit Command Chaining

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'EA'	Instruction Byte gemäß [ISO/IEC 7816-13]
P1	'00'	–
P2	'00'	–
Data	'XX…XX'	*cmdData*

[<=]

14.2.5.2 Use Case Anlegen neues Objekt, Ende der Kommandokette

Diese Variante ist zu wählen, wenn die Datenmenge in einer Kommando-APDU transferierbar ist, oder das letzte Kommando einer Chaining-Kette zu übertragen ist. In dieser Variante enthält die APDU des LOAD APPLICATION-Kommandos einen Parameter.

A_15976 - (N038.700) K_externeWelt {K_Karte}
Der Parameter *cmdData* enthält Daten, welche das neu anzulegende File beschreiben. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem, herstellerspezifischem Inhalt. Die Länge von *cmdData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.**[<=]**

A_15977 - (N038.800) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_253 verwendet werden.

Tabelle 115: CosT_253: *LOAD APPLICATION* ohne Command Chaining

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'EA'	Instruction Byte gemäß [ISO/IEC 7816-13]
P1	'00'	–
P2	'00'	–
Data	'XX…XX'	*cmdData*

[<=]

14.2.5.3 Antwort der Karte auf Anlegen neues Objekt

Tabelle 116: CosT_faf: *LOAD APPLICATION* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Ladevorgang

Tabelle 117: CosT_df0: *LOAD APPLICATION* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'6A 84'	OutOfMemory	Zu wenig Speicherplatz für neues Objekt
'6A 89'	DuplicatedObject	Identifier des neuen Objektes wird bereits verwendet
'6A 8A'	DfNameExists	AID des neuen Objektes wird bereits verwendet
'6D 00'	InstructionNotSupported	Die Karte befindet sich im Zustand "Termination state"

Hinweis CosH_c44: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_15978 - (N038.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das LOAD APPLICATION-Kommando zusätzliche Trailer verwendet.**[<=]**

14.2.5.4 Kommandoabarbeitung innerhalb der Karte

G2_N039.000.a - (N039.000)a K_COS
Das COS MUSS die LOAD APPLICATION-Varianten aus
- 14.2.5.1- Use Case Anlegen neues Objekt, nicht Ende der Kommandokette und
- 14.2.5.2- Use Case Anlegen neues Objekt, Ende der Kommandokette

unterstützen.**[<=]**

A_15979 - (N039.000)b K_TST

~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere LOAD APPLICATION-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_15980 - (N039.010) K_TST
Wenn das Attribut lifeCycleStatus des Objektsystems (siehe (N019.900)i) den Wert „Termination state" besitzt, dann MUSS es für die funktionale Zulassung zulässig sein, dass ~~KANN~~ das Kommando mit dem Trailer InstructionNot-Supported terminieren.**[<=]**

G2_N039.100 - (N039.100) K_COS
Als *affectedObject* MUSS *channelContext*. *currentFolder* verwendet werden.**[<=]**

G2_N039.200 - (N039.200) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_15981 - (N039.300) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass ein Kommando oder eine eventuell aktive Chaining-Kette wegen einer der folgenden Fälle akzeptiert oder abgelehnt wird: Das neu anzulegende File ist

a. ein Ordner und besitzt ein Attribut *applicationIdentifier* und dieser *applicationIdentifier* ist bereits einem anderen Ordner innerhalb des Objektsystems zugeordnet.
b. ein Ordner und besitzt ein Attribut *fileIdentifier* besitzt und dieser *fileIdentifier* ist bereits einem anderen File innerhalb von *affectedObject* zugeordnet.
c. eine Datei und deren Attribut *fileIdentifier* ist bereits einem anderen File innerhalb von *affectedObject* zugeordnet.
d. eine Datei und deren Attribut *shortFileIdentifier* ist bereits einer anderen Datei innerhalb von *affectedObject* zugeordnet.

[<=]

G2_N039.300.a - (N039.300)a K_COS
Wenn das Kommando abgelehnt wird, weil das neu anzulegende File ein Ordner ist und ein Attribut *applicationIdentifier* besitzt und dieser *applicationIdentifier* ist bereits einem anderen Ordner innerhalb des Objektsystems zugeordnet, genau dann MUSS das Kommando mit dem Trailer DfNameExists terminieren.**[<=]**

G2_N039.300.b - (N039.300)b K_COS
Wenn das Kommando abgelehnt wird, weil das neu anzulegende File ein Ordner ist und ein Attribut *fileIdentifier* besitzt und dieser *fileIdentifier* ist bereits einem anderen File in *affectedObject* zugeordnet, dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren.**[<=]**

G2_N039.300.c - (N039.300)c K_COS
Wenn das Kommando abgelehnt wird, weil das neu anzulegende File eine Datei ist, deren Attribut *fileIdentifier* bereits einem anderen File in *affectedObject* zugeordnet ist, dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren.**[<=]**

G2_N039.300.d - (N039.300)d K_COS
Wenn das Kommando abgelehnt wird, weil das neu anzulegende File eine Datei ist und ein Attribut *shortFileIdentifier* besitzt und dieser *shortFileIdentifier* ist bereits einer anderen

Datei in *affectedObject* zugeordnet, dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren.**[<=]**

G2_N039.400.a - (N039.400)a K_COS
Das Kommando MUSS akzeptiert werden, wenn das neu anzulegende Objekt ein Ordner ist und ein Attribut *applicationIdentifier* besitzt und dieser *applicationIdentifier* keinem anderen Ordner innerhalb des Objektsystems zugeordnet ist.**[<=]**

G2_N039.400.b - (N039.400.)b K_COS
Das Kommando MUSS akzeptiert werden, wenn das neu anzulegende Objekt ein Ordner ist und ein Attribut *fileIdentifier* besitzt und dieser *fileIdentifier* keinem anderen File innerhalb von *affectedObject* zugeordnet ist.**[<=]**

G2_N039.400.c - (N039.400)c K_COS
Das Kommando MUSS akzeptiert werden, wenn das neu anzulegende Objekt eine Datei ist, deren Attribut *fileIdentifier* keinem anderen File innerhalb von *affectedObject* zugeordnet ist.**[<=]**

G2_N039.400.d - (N039.400)d K_COS
Das Kommando MUSS akzeptiert werden, wenn das neu anzulegende Objekt eine Datei ist und ein Attribut *shortFileIdentifier* besitzt und dieser *shortFileIdentifier* keiner anderen Datei innerhalb von *affectedObject* zugeordnet ist.**[<=]**

G2_N039.500 - (N039.500) K_COS
Wenn insgesamt ausreichender, aber nicht genügend zusammenhängender Speicherplatz vorhanden ist, dann MUSS das COS intern dafür sorgen, dass diese Operation trotzdem erfolgreich durchführbar ist. Typischerweise wird diese Operation als "Defragmentieren" bezeichnet.**[<=]**

G2_N039.600 - (N039.600) K_COS
Wenn nicht genügend Speicherplatz zum Anlegen des neuen Objektes vorhanden ist, genau dann MUSS das Kommando mit dem Trailer OutOfMemory terminieren.**[<=]**

A_15982 - (N039.700) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_15983 - (N039.800) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N039.900 - (N039.900) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_15984 - (N040.000) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_df0 ist herstellerspezifisch.
b. Jeder Trailer in CosT_df0 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

G2_N040.100.a - (N040.100)a K_COS
Wenn dies das einzige oder letzte Kommando einer Command-Chaining-Kette ist und das neu angelegte Objekt ist ein Ordner, dann MUSS im Erfolgsfall *channelContext.currentFolder* auf den neu angelegten Ordner gesetzt werden. Dabei sind die Regeln zum

Setzen des Kanalkontextes für den neuen Ordner zu berücksichtigen (siehe (N048.200)b).**[<=]**

G2_N040.100.b - (N040.100)b K_COS
Wenn dies das einzige oder letzte Kommando einer Command-Chaining-Kette ist und das neu angelegte Objekt ist eine Datei, dann MUSS im Erfolgsfall *currentEF* auf die neu angelegte Datei gesetzt werden.**[<=]**

G2_N040.200.a - (N040.200)a K_COS
Wenn das Kommando nicht erfolgreich verlief, dann DÜRFEN *currentFolder* und *currentEF* NICHT verändert werden, auch wenn ein LOAD APPLICATION-Kommando mit einem Trailer aus CosT_df0 terminierte, sondern MÜSSEN denselben Wert besitzen, wie vor der Ausführung dieses LOAD APPLICATION-Kommandos.**[<=]**

G2_N040.200.b - (N040.200)b K_COS
Wenn das Kommando nicht erfolgreich verlief, dann DÜRFEN *currentFolder* und *currentEF* NICHT verändert werden, auch wenn eine Command-Chaining-Kette abgebrochen wird (siehe (N029.874)), sondern MÜSSEN denselben Wert besitzen, wie vor dem Start dieser Command-Chaining-Kette.**[<=]**

G2_N040.300 - (N040.300) K_COS
Wenn dieses LOAD APPLICATION-Kommando mit einem Trailer aus CosT_df0 terminierte, dann MUSS ein eventuell aktives Command Chaining abgebrochen werden.**[<=]**

A_15985 - (N040.400) K_COS
Das neu angelegte bzw. neu anzulegende Objekt MUSS inklusive Freigabe eines eventuell von ihm allokierten Speichers aus dem Objektsystem gelöscht werden (komplettes Roll-Back der Chaining-Kette), wenn**[<=]**

G2_N040.400.a - (N040.400)a K_COS
das LOAD APPLICATION-Kommando mit einem Trailer aus CosT_df0 terminierte.**[<=]**

G2_N040.400.b - (N040.400)b K_COS
das LOAD APPLICATION-Kommando während der Kommandobearbeitung durch einen Reset abgebrochen wird.**[<=]**

G2_N040.400.c - (N040.400)c K_COS
eine Command-Chaining-Kette abgebrochen wird (siehe (N029.874)).**[<=]**

Hinweis CosH_27f: Es ist denkbar, dass Anforderung (N039.500) wie folgt getestet wird:

3. *Ausgangspunkt sei eine Smartcard, deren Objektsystem nur eine sehr geringe (minimale) Anzahl von Objekten enthält.*
4. *Schritt 1: Per LOAD APPLICATION-Kommando werde eine Datei (transparent oder strukturiert wird zufällig bestimmt) mit 200 Oktett Nutzdaten angelegt.*
5. *Schritt 2: Schritt 1 wird so lange wiederholt, bis das LOAD APPLICATION-Kommando mit dem Trailer OutOfMemory terminiert.*
6. *Schritt 3: Es werden zwei der zuvor angelegten Dateien zufällig ausgewählt und gelöscht (DELETE-Kommando). Anschließend werde per LOAD APPLICATION eine neue Datei (transparent oder strukturiert wird zufällig bestimmt) angelegt. Wenn die Summe der Nutzdaten der in diesem Schritt gelöschten Dateien x ist, dann werde als Größe der Nutzdaten der in diesem Schritt neu angelegten Datei ebenfalls x gewählt. Es wird erwartet, dass dieses LOAD APPLICATION-Kommando nicht mit OutOfMemory terminiert.*

7. *Schritt 4: Schritt 3 werde so lange wiederholt, bis nur noch eine Datei übrig ist, welche durch diesen Algorithmus angelegt wurde, oder nur noch Dateien übrig sind, welche die maximal mögliche Dateigröße gemäß (N011.500) bzw. (N013.000) besitzen. Zwar bezieht sich (N013.000) nur auf linear variable EF, ist aber analog übertragbar auf andere lineare EF.*

14.2.6 SELECT

Das Kommando SELECT sucht im Objektsystem nach einem File (Ordner oder Datei) und wählt dieses aus. Das Auswählen ist vielfach Voraussetzung, damit andere Use Cases (Lesen, Schreiben, …) erfolgreich durchführbar sind. Optional ist es möglich, in den Antwortdaten die wesentlichen Attribute des Files zurückzumelden. Welches File selektiert wird, bestimmen Parameter in der Kommandonachricht.

14.2.6.1 Use Case Selektieren ohne AID, first, keine Antwortdaten

Diese Variante selektiert das Wurzelverzeichnis des Objektsystems. In dieser Variante enthält die APDU des SELECT-Kommandos drei Parameter:

A_15986 - (N040.500) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_15987 - (N040.600) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_15988 - (N040.700) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.**[<=]**

A_15989 - (N040.800) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_d79 verwendet werden.

Tabelle 118: CosT_d79: *SELECT*, kein AID, first occurrence, keine Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer)
P2	'0C'	*fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten

[<=]

14.2.6.2 Use Case Selektieren ohne AID, first, Antwortdaten mit FCP

Diese Variante selektiert das Wurzelverzeichnis des Objektsystems. In dieser Variante enthält die APDU des SELECT-Kommandos vier Parameter:

A_15990 - (N040.900) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_15991 - (N041.000) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_15992 - (N041.100) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.**[<=]**

A_15993 - (N041.200) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_15994 - (N041.300) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_4c6 verwendet werden.

Tabelle 119: CosT_4c6: *SELECT*, kein AID, first occurrence, Antwortdaten mit FCP

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer)
P2	'04'	*fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.3 Use Case Selektieren ohne AID, next, keine Antwortdaten

Der wiederholte Aufruf dieser Variante selektiert nacheinander alle Ordner, die das Attribut *applicationIdentifier* besitzen (Applikationen gemäß 8.3.1.1 und ADF gemäß 8.3.1.3). In dieser Variante enthält die APDU des SELECT-Kommandos drei Parameter:

A_15995 - (N041.400) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_15996 - (N041.500) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '2' gewählt werden.**[<=]**

A_15997 - (N041.600) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.**[<=]**

A_15998 - (N041.700) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_b77 verwendet werden.

Tabelle 120: CosT_b77: *SELECT*, kein AID, next occurrence, keine Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer)
P2	'0E'	*fileOccurrence* + *responseType* = next occurrence, keine Antwortdaten

[<=]

14.2.6.4 Use Case Selektieren ohne AID, next, Antwortdaten mit FCP

Der wiederholte Aufruf dieser Variante selektiert nacheinander alle Ordner, die das Attribut *applicationIdentifier* besitzen (Applikationen gemäß 8.3.1.1 und ADF gemäß 8.3.1.3). In dieser Variante enthält die APDU des SELECT-Kommandos vier Parameter:

A_15999 - (N041.800) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_16000 - (N041.900) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '2' gewählt werden.**[<=]**

A_16001 - (N042.000) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.**[<=]**

A_16002 - (N042.100) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16003 - (N042.200) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_cfb verwendet werden.

Tabelle 121: CosT_cfb: *SELECT*, kein AID, next occurrence, Antwortdaten mit FCP

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier* (hier leer)
P2	'06'	*fileOccurrence* + *responseType* = next occurrence, FCP-Antwortdaten
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.5 Use Case Selektieren per AID, first, keine Antwortdaten

In dieser Variante enthält die APDU des SELECT-Kommandos vier Parameter:

A_16004 - (N042.300) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_16005 - (N042.400) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_16006 - (N042.500) K_externeWelt {K_Karte}
Der Parameter *aid* MUSS einen Oktettstring gemäß (N010.200) oder dessen Anfang enthalten. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.**[<=]**

A_16007 - (N042.600) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.**[<=]**

A_16008 - (N042.700) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_e60 verwendet werden.

Tabelle 122: CosT_e60: *SELECT*, AID, first occurrence, keine Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit applicationIdentifier
P2	'0C'	*fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten
Data	'XX…XX'	*aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16]

[<=]

14.2.6.6 Use Case Selektieren per AID, first, Antwortdaten mit FCP

In dieser Variante enthält die APDU des SELECT-Kommandos fünf Parameter:

A_16009 - (N042.800) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_16010 - (N042.900) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_16011 - (N043.000) K_externeWelt {K_Karte}
Der Parameter *aid* MUSS einen Oktettstring gemäß (N010.200) oder dessen Anfang enthalten. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.**[<=]**

A_16012 - (N043.100) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.**[<=]**

A_16013 - (N043.200) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16014 - (N043.300) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_18c verwendet werden.

Tabelle 123: CosT_18c: SELECT, AID, first occurrence, Antwortdaten mit FCP

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier*
P2	'04'	*fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP
Data	'XX…XX'	*aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16]
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.7 Use Case Selektieren per AID, next, keine Antwortdaten

In dieser Variante enthält die APDU des SELECT-Kommandos vier Parameter:

A_16015 - (N043.400) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_16016 - (N043.500) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '2' gewählt werden.**[<=]**

A_16017 - (N043.600) K_externeWelt {K_Karte}
Der Parameter *aid* enthält einen Oktettstring gemäß (N010.200) oder dessen Anfang. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.**[<=]**

A_16018 - (N043.700) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.**[<=]**

A_16019 - (N043.800) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_776 verwendet werden.

Tabelle 124: CosT_776: *SELECT*, AID, next occurrence, keine Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier*
P2	'0E'	*fileOccurrence* + *responseType* = next occurrence, keine Antwortdaten
Data	'XX…XX'	*aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16]

[<=]

14.2.6.8 Use Case Selektieren per AID, next, Antwortdaten mit FCP

In dieser Variante enthält die APDU des SELECT-Kommandos fünf Parameter:

A_16020 - (N043.900) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_16021 - (N044.000) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '2' gewählt werden.**[<=]**

A_16022 - (N044.100) K_externeWelt {K_Karte}
Der Parameter *aid* enthält einen Oktettstring gemäß (N010.200) oder dessen Anfang. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.**[<=]**

A_16023 - (N044.200) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.**[<=]**

A_16024 - (N044.300) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16025 - (N044.400) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_203 verwendet werden.

Tabelle 125: CosT_203: *SELECT*, AID, next occurrence, Antwortdaten mit FCP

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier*
P2	'06'	*fileOccurrence* + *responseType* = next occurrence,Antwortdaten mit FCP
Data	'XX…XX'	*aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16]
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.9 Use Case Selektieren, DF oder ADF, keine Antwortdaten

In dieser Variante enthält die APDU des SELECT-Kommandos vier Parameter:

A_16026 - (N044.500) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '01' gewählt werden.**[<=]**

A_16027 - (N044.600) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_16028 - (N044.700) K_externeWelt {K_Karte}
Der Parameter *fid* MUSS einen Oktettstring gemäß (N006.700) enthalten. Im *currentFolder* wird nach einem Ordner mit dazu passendem Attribut *fileIdentifier* gesucht.**[<=]**

A_16029 - (N044.800) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.**[<=]**

A_16030 - (N044.900) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_28f verwendet werden.

Tabelle 126: CosT_28f: *SELECT*, DF oder ADF mit FID, keine Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'01'	*selectionMode* = Selektion eines Ordners mit *fileIdentifier*
P2	'0C'	*fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten
Data	'XXXX'	*fid*

[<=]

14.2.6.10 Use Case Selektieren, DF oder ADF, Antwortdaten mit FCP

In dieser Variante enthält die APDU des SELECT-Kommandos fünf Parameter:

A_16031 - (N045.000) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '01' gewählt werden.**[<=]**

A_16032 - (N045.100) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_16033 - (N045.200) K_externeWelt {K_Karte}
Der Parameter *fid* MUSS einen Oktettstring gemäß (N006.700) enthalten. Im *currentFolder* wird nach einem Ordner mit dazu passendem Attribut *fileIdentifier* gesucht.**[<=]**

A_16034 - (N045.300) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.**[<=]**

A_16035 - (N045.400) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16036 - (N045.500) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_19d verwendet werden.

Tabelle 127: CosT_19d: *SELECT*, DF oder ADF mit FID, Antwortdaten mit FCP

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'01'	*selectionMode* = Ordnerselektion mit *fileIdentifier*
P2	'04'	*fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP
Data	'XXXX'	*fid*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.11 Use Case Selektieren übergeordnetes Verzeichnis ohne FCP

Diese Variante selektiert den eine Ebene höher liegenden Ordner (siehe (N020.000), (N020.100)). In dieser Variante enthält die APDU des SELECT-Kommandos zwei Parameter:

A_16037 - (N045.600) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '03' gewählt werden.**[<=]**

A_16038 - (N045.700) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.**[<=]**

A_16039 - (N045.800) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_0e5 verwendet werden.

Tabelle 128: CosT_0e5: *SELECT*, höhere Ebene keine Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'03'	*selectionMode* = Selektion des eine Ebene höheren Ordners
P2	'0C'	*responseType* = keine Antwortdaten

[<=]

14.2.6.12 Use Case Selektieren übergeordnetes Verzeichnis mit FCP

Diese Variante selektiert den eine Ebene höher liegenden Ordner (siehe (N020.000), (N020.100)). In dieser Variante enthält die APDU des SELECT-Kommandos drei Parameter:

A_16040 - (N045.900) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '03' gewählt werden.**[<=]**

A_16041 - (N046.000) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.**[<=]**

A_16042 - (N046.100) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16043 - (N046.200) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_e82 verwendet werden.

Tabelle 129: CosT_e82: *SELECT*, höhere Ebene, Antwortdaten mit FCP

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'03'	*selectionMode* = Selektion des eine Ebene höheren Ordners
P2	'04'	*responseType* = Antwortdaten mit FCP
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.13 Use Case Selektieren einer Datei, keine Antwortdaten

In dieser Variante enthält die APDU des SELECT-Kommandos vier Parameter:

A_16050 - (N046.300) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '02' gewählt werden.**[<=]**

A_16051 - (N046.400) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_16052 - (N046.500) K_externeWelt {K_Karte}
Der Parameter *fid* MUSS einen Oktettstring gemäß (N006.700) enthalten. Im *currentFolder* wird nach einer Datei mit dazu passendem Attribut *fileIdentifier* gesucht.**[<=]**

A_16053 - (N046.600) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.**[<=]**

A_16054 - (N046.700) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_e0d verwendet werden.

Tabelle 130: CosT_e0d: *SELECT*, EF mit FID, keine Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'02'	*selectionMode* = Selektion einer Datei mit *fileIdentifier*
P2	'0C'	*fileOccurrence* + *responseType* = first occurrence, keine Antwortdaten
Data	'XXXX'	*fid*

[<=]

14.2.6.14 Use Case Selektieren einer Datei, Antwortdaten mit FCP

In dieser Variante enthält die APDU des SELECT-Kommandos fünf Parameter:

A_16055 - (N046.800) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '02' gewählt werden.**[<=]**

A_16056 - (N046.900) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_16057 - (N047.000) K_externeWelt {K_Karte}
Der Parameter *fid* enthält einen Oktettstring gemäß (N006.700). Im *currentFolder* wird nach einer Datei mit dazu passendem Attribut *fileIdentifier* gesucht.**[<=]**

A_16058 - (N047.100) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.**[<=]**

A_16059 - (N047.200) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16060 - (N047.300) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_55b verwendet werden.

Tabelle 131: CosT_55b: SELECT, EF mit FID, Antwortdaten mit FCP

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'02'	*selectionMode* = Selektion einer Datei mit *fileIdentifier*
P2	'04'	*fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCP
Data	'XXXX'	*fid*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.15 Use Case Selektieren per AID, first, Antwortdaten mit FCI

In dieser Variante enthält die APDU des SELECT-Kommandos fünf Parameter:

A_18357 - (N047.350) K_externeWelt {K_Karte}
Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.**[<=]**

A_18358 - (N047.352) K_externeWelt {K_Karte}
Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passenden Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.**[<=]**

A_18359 - (N047.354) K_externeWelt {K_Karte}
Der Parameter *aid* MUSS einen Oktettstring gemäß (N010.200) oder dessen Anfang enthalten. Im Objektsystem wird nach einem Ordner mit dazu passendem Attribut *applicationIdentifier* gesucht.**[<=]**

A_18360 - (N047.356) K_externeWelt {K_Karte}
Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '00' gewählt werden.**[<=]**

A_18361 - (N047.358) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich *WildCardShort* sein.**[<=]**

A_18362 - (N047.360) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_812 verwendet werden.

Tabelle 132: CosT_812: SELECT, AID, first occurrence, Antwortdaten mit FCI

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'04'	*selectionMode* = Ordnerselektion mit *applicationIdentifier*
P2	'00'	*fileOccurrence* + *responseType* = first occurrence, Antwortdaten mit FCI
Data	'XX…XX'	*aid*, Oktettstring, Anzahl Oktette aus dem Intervall [1, 16]
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.2.6.16 Zusammenfassung der SELECT-Kommando-Varianten

Wegen der Vielzahl an Varianten für dieses Kommando werden hier alle auf einen Blick dargestellt. Es sei darauf hingewiesen, dass nicht alle Kombinationen der folgenden Tabelle in den vorangegangenen Kapiteln enthalten sind. Deshalb sind solche nicht zwingend zu unterstützen.

Tabelle 133: CosT_59c: *SELECT*, Kommandoparameter im Überblick

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'01' '02' '03' '04'	Selektion eines Ordners mit *fileIdentifier* Selektion einer Datei mit *fileIdentifier* Selektion des eine Ebene höher liegenden Ordners Selektion eines Ordners mit (möglicherweise leerem) *applicationIdentifier*
P2	'00' '04' '06' '0C' '0E'	first occurrence, Antwortdaten mit FCI first occurrence, Antwortdaten mit FCP next occurrence, Antwortdaten mit FCP first occurrence, keine Antwortdaten next occurrence, keine Antwortdaten
Data	'XX…XX'	P1 = '01': zwei Byte *fid* P1 = '02': zwei Byte *fid* P1 = '03': abwesend P1 = '04': abwesend, oder bis zu 16 Oktette *aid*
Le	*length*	P2 = '04': Anzahl der erwarteten Oktette in den Antwortdaten P2 = '06': Anzahl der erwarteten Oktette in den Antwortdaten P2 = '0C': abwesend P2 = '0E': abwesend

14.2.6.17 Antwort der Karte auf Selektieren eines Files

Tabelle 134: CosT_906: *SELECT* Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	FCP	Abwesend, oder File Control Parameter
Trailer	**Inhalt**	**Beschreibung**
'62 83'	FileDeactivated	Selektiertes File ist logisch oder physikalisch deaktiviert
'62 85'	FileTerminated	Selektiertes File ist logisch oder physikalisch terminiert
'90 00'	NoError	Erfolgreiche Selektion eines Files

Tabelle 135: CosT_0dc: *SELECT* Antwort-APDU im Fehlerfall

Trailer	Inhalte	Beschreibung
'6A 82'	FileNotFound	Zu selektierendes File wurde nicht gefunden
'6D 00'	InstructionNotSupported	Die Karte befindet sich im Zustand "Termination state"

Hinweis CosH_237: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16061 - (N047.400) K_TST
Die funktionale Eignung DARF einem Prüfling NICHT mit der Begründung verwehrt werden das Kommando SELECT verwende zusätzliche Trailer.**[<=]**

14.2.6.18 Kommandoabarbeitung innerhalb der Karte

Zur Beschreibung der Kommandobearbeitung werden folgende Definitionen eingeführt:

Tabelle 136: CosT_e2e: Innerhalb der Beschreibung des *SELECT*-Kommandos verwendete Definitionen

Entität	Beschreibung
oldFolder	Dies ist die Bezeichnung von *currentFolder* vor Ausführung dieses SELECT-Kommandos
newFile	Dies ist die Bezeichnung für das File (Ordner oder Datei), welches im Rahmen der Selektion ausgewählt wird
path(*folder*)	Pfad eines Ordners mit Namen *folder*. Zum Pfad gehören nach dieser Definition sowohl der Ordner *folder* als auch alle seine übergeordneten Verzeichnisse einschließlich *root*.

G2_N047.500.a - (N047.500)a K_COS
Das COS MUSS die SELECT-Varianten aus
- 14.2.6.1- Use Case Selektieren ohne AID, first, keine Antwortdaten,
- 14.2.6.2- Use Case Selektieren ohne AID, first, Antwortdaten mit FCP,
- 14.2.6.5- Use Case Selektieren per AID, first, keine Antwortdaten,
- 14.2.6.6- Use Case Selektieren per AID, first, Antwortdaten mit FCP,
- 14.2.6.9- Use Case Selektieren, DF oder ADF, keine Antwortdaten,
- 14.2.6.10- Use Case Selektieren, DF oder ADF, Antwortdaten mit FCP,
- 14.2.6.11- Use Case Selektieren übergeordnetes Verzeichnis ohne FCP,

- 14.2.6.12- Use Case Selektieren übergeordnetes Verzeichnis mit FCP,
- 14.2.6.13- Use Case Selektieren einer Datei, keine Antwortdaten,
- 14.2.6.14- Use Case Selektieren einer Datei, Antwortdaten mit FCP und
- 14.2.6.15- Use Case Selektieren per AID, first, Antwortdaten mit FCI

unterstützen.**[<=]**

A_16062 - (N047.500)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS die SELECT-Varianten
- 14.2.6.3- Use Case Selektieren ohne AID, next, keine Antwortdaten,
- 14.2.6.4- Use Case Selektieren ohne AID, next, Antwortdaten mit FCP,
- 14.2.6.7- Use Case Selektieren per AID, next, keine Antwortdaten und
- 14.2.6.8- Use Case Selektieren per AID, next, Antwortdaten mit FCP

sowie weitere SELECT-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N047.590 - (N047.590) K_COS
Wenn das Attribut *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) den Wert "Termination state" besitzt, genau dann MUSS das Kommando mit dem Trailer InstructionNotSupported terminieren.**[<=]**

A_16063 - (N047.600) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS im Rahmen der Bearbeitung des SELECT-Kommandos
a. Zugriffsregeln auswertet oder
b. als Zugriffsbedingung eines SELECT-Kommandos stets ALWAYS verwendet.**[<=]**

G2_N047.700 - (N047.700) K_COS
Wenn der Parameter P1 den Wert '01' besitzt, dann MUSS in *oldFolder.children* nach einem Ordner gesucht werden, der ein Attribut *fileIdentifier* besitzt, dessen Wert mit dem Parameter *fid* aus den Kommandodaten übereinstimmt. Wenn ein solcher Ordner existiert, dann MUSS *newFile* auf den gefundenen Ordner gesetzt werden. Andernfalls ist die Suche erfolglos.**[<=]**

G2_N047.800 - (N047.800) K_COS
Wenn der Parameter P1 den Wert '02' besitzt, dann MUSS in *oldFolder.children* nach einer Datei gesucht werden, deren Attribut *fileIdentifier* mit dem Parameter *fid* aus den Kommandodaten übereinstimmt. Wenn eine solche Datei existiert, dann MUSS *newFile* auf die gefundene Datei gesetzt werden. Andernfalls ist die Suche erfolglos.**[<=]**

G2_N047.900.a - (N047.900)a K_COS
Wenn der Parameter P1 den Wert '03' besitzt, und *currentFolder* ist gleich *root*, dann MUSS das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N047.900.b - (N047.900)b K_COS
Wenn der Parameter P1 den Wert '03' besitzt, und *currentFolder* ist nicht *root*, dann ist *newFile* der im Vergleich zu *currentFolder* eine Ebene höher liegende Ordner (siehe (N020.000), (N020.100)).**[<=]**

A_16064 - (N048.000) K_COS
Wenn der Parameter P1 den Wert '04' besitzt, dann enthält die Kommandonachricht einen (möglicherweise leeren) Parameter *aid*. In diesem Fall ist folgender Algorithmus abzuarbeiten:**[<=]**

G2_N048.000.a - (N048.000)a K_COS

Schritt 1: Im gesamten Objektsystem wird nach Ordnern gesucht, die ein Attribut *applicationIdentifier* besitzen, welches zu *aid* passt. Ein Attribut *applicationIdentifier* MUSS als passend betrachtet werden, wenn
1. *applicationIdentifier* identisch zu *aid* ist.
2. *aid* leer ist.**[<=]**

A_16065 - (N048.000)a.3 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass die Most Significant Byte eines Attributes *applicationIdentifier* mit *aid* übereinstimmen und dies als
i. passend betrachtet wird.
ii. unpassend betrachtet wird.**[<=]**

G2_N048.000.b - (N048.000)b K_COS
Schritt 2: Alle passenden Ordner werden zu einer Liste zusammengestellt.
1. Ist *root* in dieser Liste enthalten, dann MUSS es das erste Listenelement sein.
2. Solange das Objektsystem nicht verändert wird (DELETE oder LOAD APPLICATION) MUSS bei identischem *aid* stets dieselbe Liste erstellt werden.**[<=]**

G2_N048.000.c - (N048.000)c K_COS
Schritt 3.1: Hat P2 einen Wert aus der Menge {'04', '0C'}, dann MUSS *newFile* auf das erste Listenelement gesetzt werden.**[<=]**

A_16066 - (N048.000)d K_COS
Schritt 3.2: Hat P2 einen Wert aus der Menge {'06', '0E'} und
1. *oldFolder* ist ebenfalls in der Liste und *oldFolder* ist
   i. nicht das letzte Listenelement, dann MUSS *newFile* auf das nächste Listenelement gesetzt werden.
   ii. das letzte Listenelement, dann MUSS die Suche erfolglos sein.
2. *oldFolder* ist nicht in der Liste enthalten, dann MUSS entweder
   i. *newFile* auf irgendein Listenelement gesetzt werden, oder
   ii. die Suche erfolglos sein.**[<=]**

G2_N048.100 - (N048.100) K_COS
Wenn die Suche erfolglos war, genau dann MUSS das Kommando mit dem Trailer FileNotFound terminieren. Dabei DÜRFEN *currentFolder*, *currentEF* sowie der Kanalkontext (siehe 12.1) NICHT verändert werden.**[<=]**

G2_N048.200.a - (N048.200)a K_COS
Wenn *newFile* eine Datei ist, dann MUSS *currentEF* auf *newFile* gesetzt werden.**[<=]**

A_16067 - (N048.200)b K_COS
Wenn *newFile* ein Ordner ist, dann MUSS folgender Algorithmus abgearbeitet werden:**[<=]**

G2_N048.200.b.1 - (N048.200)b.1 K_COS
Schritt 1: *currentFolder* MUSS gleich *newFile* gesetzt werden.**[<=]**

G2_N048.200.b.2 - (N048.200)b.2 K_COS
Schritt 2: *currentEF* MUSS auf den Wert "undefiniert" gesetzt werden.**[<=]**

G2_N048.200.b.3 - (N048.200)b.3 K_COS
Schritt 3: In allen Ordnern, die sowohl zu path(*oldFolder*) als auch zu path(*newFile*) gehören, MUSS *seIdentifier* unverändert bleiben.**[<=]**

G2_N048.200.b.4 - (N048.200)b.4 K_COS
Schritt 4: In allen anderen Ordnern MUSS *seIdentifier* auf den Wert 1 gesetzt werden.**[<=]**

G2_N048.200.b.5 - (N048.200)b.5 K_COS
Schritt 5: Die Funktion clearSecurityStatusFolder(*folder*) MUSS für alle Ordner ausgeführt werden, die nicht zu path(*newFile*) gehören.**[<=]**

G2_N048.200.b.6 - (N048.200)b.6 K_COS, Option_Kryptobox
Schritt 6: Die Funktion clearSessionkeys( ) MUSS ausgeführt werden, falls *SessionkeyContext. folderSessionkeys* nicht zu path(*newFile*) gehört.**[<=]**

G2_N048.200.b.7 - (N048.200)b.7 K_COS
Schritt 7: Aus *dfSpecificPasswordList* MÜSSEN mittels clearPasswordStatus(…) alle Einträge entfernt werden, die nicht zu path(*newFile*) gehören.**[<=]**

G2_N048.200.b.8 - (N048.200)b.8 K_COS
Schritt 8: Jedes Element von *keyReferenceList*, welches ein Schlüsselobjekt referenziert, das zu einem Ordner außerhalb von path(*newFile*) gehört, MUSS auf den Wert "leer" gesetzt werden.**[<=]**

G2_N048.300.a - (N048.300).a K_COS
Wenn P2 einen Wert aus der Menge {'04', '06'} hat, genau dann MUSS das Datenfeld *rspData* der Antwortnachricht die File Control Parameter gemäß 8.3.3 wie folgt enthalten: Sei FCP ein Oktettstring, der die File Control Parameter gemäß 8.3.3 enthält, dann gilt: Wenn OctetLength(FCP) kleiner Nr gemäß (N027.200),
1. dann *rspData* = FCP.
2. sonst *rspData* = Extract_MSByte(FCP, Nr).**[<=]**

G2_N048.300.b - (N048.300)b K_COS
Wenn P2 einen Wert aus der Menge {'0C, '0E'} hat, dann MUSS das Datenfeld *rspData* der Antwortnachricht fehlen.**[<=]**

A_18356 - (N048.300)c K_COS
Wenn P2 einen Wert aus der Menge {'00"} hat, genau dann MUSS das Datenfeld *rspData* der Antwortnachricht die File Control Information (FCI) wie folgt enthalten:

1. Die File Control Information (FCI) sind ein DER-codiertes Datenobjekt mit Tag '6F'.
2. Die Anzahl der Oktette im Datenfeld *rspData* der Antwortnachricht ist kleiner gleich 256 (short length und FCI vollständig in *rspData* enthalten).
3. Die FCI enthalten mindestens ein DO'84' mit dem Attribut *applicationIdentifier*, dessentwegen dieser Ordner ausgewählt wurde. Es ist zulässig, dass die FCI mehrere DO'84' enthalten.
4. Die FCI enthalten ein DO'A5', für das gilt:
   a. Das DO'A5' enthält ein DO'9F65'. Das Wertfeld des DO'9F65' enthält eine Zahl*maxLc* in zwei Oktetten gemäß I2OS(*maxLc*, 2). Die Zahl *maxLc* ist größer gleich 1024 (vergleiche (N029.890)a.3) und kleiner als die maximal vom COS unterstützte Länge einer geschützten Kommandonachricht.
   b. Es ist zulässig, dass das DO'A5' stets wie folgt codiert wird: 'A5 – 05 – (9F65 – 02 – 0400) '
   c. Es ist zulässig, dass das DO'A5' weitere Datenobjekte enthält.
5. Es ist zulässig, dass FCI weitere Datenobjekte enthält.

[<=]

G2_N048.400.a - (N048.400)a K_COS
Wenn der logische Wert von *newFile.lifeCycleStatus* (siehe (N020.600)) den Wert "Operational state (deactivated)" hat, genau dann MUSS als Trailer FileDeactivated gewählt werden.**[<=]**

G2_N048.400.b - (N048.400)b K_COS
Wenn der logische Wert von *newFile.lifeCycleStatus* (siehe (N020.600)) den Wert "Termination state" hat, genau dann MUSS als Trailer FileTerminated gewählt werden.**[<=]**

G2_N048.500 - (N048.500) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16069 - (N048.600)a K_TST
Die Priorität der Trailer in CosT_0dc MUSS herstellerspezifisch sein.**[<=]**

G2_N048.600.b - (N048.600)b K_COS
Jeder Trailer in CosT_0dc MUSS eine höhere Priorität als FileDeactivated haben.**[<=]**

G2_N048.600.c - (N048.600)c K_COS
FileDeactivated MUSS eine höhere Priorität als NoError haben.**[<=]**

Hinweis CosH_213: Gemäß den Regeln dieses Dokumentes ist es zulässig, in deaktivierten oder terminierten Ordnern vorhandene Unterordner oder Dateien mittels SELECT*-Kommando zu selektieren.*

14.2.7 TERMINATE CARD USAGE

Das Kommando TERMINATE CARD USAGE überführt eine Karte irreversibel in den Zustand "Termination state". Das Kommando ist unabhängig vom aktuellen Wert von*currentFolder* und *currentEF* ausführbar. Während der Kommandoausführung wird im Objektsystem das Attribut *lifeCycleStatus* auf den Wert "Termination state" gesetzt und, mit Ausnahme des Basiskanals, werden alle weiteren logischen Kanäle geschlossen. Anschließend ist die Funktionalität des SELECT-Kommandos nicht mehr verfügbar (siehe (N047.590)).

14.2.7.1 Use Case Terminieren der Karte

In dieser Variante enthält die APDU des TERMINATE CARD USAGE-Kommandos keinen Parameter:

A_16070 - (N048.700) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_6fa verwendet werden.

Tabelle 137: CosT_6fa: *TERMINATE CARD USAGE*

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'FE'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	–
P2	'00'	–

[<=]

14.2.7.2 Antwort der Karte auf Terminieren der Karte

Tabelle 138: CosT_fc1: *TERMINATE CARD USAGE* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Terminierung

Tabelle 139: CosT_d53: *TERMINATE CARD USAGE* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis CosH_0b8: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16071 - (N048.738) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das TERMINATE CARD USAGE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.2.7.3 Kommandoabarbeitung innerhalb der Karte

G2_N048.740.a - (N048.740)a K_COS
Das COS MUSS die TERMINATE CARD USAGE-Variante aus
- 14.2.7.1- Use Case Terminieren der Karte
unterstützen.**[<=]**

A_16072 - (N048.740)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere TERMINATE CARD USAGE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N048.742 - (N048.742) K_COS
affectedObject MUSS gleich *root* gesetzt werden.**[<=]**

G2_N048.744 - (N048.744) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N048.748 - (N048.748) K_COS
Wenn die Kommandos-APDU dieses TERMINATE CARD USAGE-Kommandos gemäß 13.2 mittels Sessionkeys gesichert war, dann MUSS die korrespondierende Antwort-APDU gemäß 13.3 mit denselben Sessionkeys gesichert sein, auch wenn als Seiteneffekt von (N048.756)b oder (N048.756)c Sessionkeys gelöscht werden.**[<=]**

G2_N048.752 - (N048.752) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* des Objektsystems den Wert "Termination state" besitzt, dann MUSS als Trailer NoError verwendet werden.**[<=]**

G2_N048.756.a - (N048.756)a K_COS
Der physikalische Wert von *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) MUSS mittels Transaktionsschutz auf den Wert "Termination state" gesetzt werden.**[<=]**

G2_N048.756.b - (N048.756)b K_COS
Bis auf den Basiskanal MÜSSEN alle anderen logischen Kanäle geschlossen werden.**[<=]**

G2_N048.756.c - (N048.756)c K_COS
Im Basiskanal MUSS *channelContext* gemäß (N030.100) gesetzt werden.**[<=]**

A_16073 - (N048.760) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16074 - (N048.762) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N048.764 - (N048.764) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16075 - (N048.766) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_d53 ist herstellerspezifisch.
b. Jeder Trailer in CosT_d53 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.2.8 TERMINATE DF

Das Kommando TERMINATE DF überführt einen Ordner irreversibel in den Zustand "Termination state". Der betroffene Ordner wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses TERMINATE DF-Kommandos durch eine Select-Operation (SELECT-Kommandos).

14.2.8.1 Use Case Terminieren eines Ordners

In dieser Variante enthält die APDU des TERMINATE DF-Kommandos keinen Parameter:

A_16079 - (N048.800) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_f82 verwendet werden.

Tabelle 140: CosT_f82: *TERMINATE DF* aktueller Ordner

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E6'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	–
P2	'00'	–

[<=]

14.2.8.2 Antwort der Karte auf Terminieren eines Ordners

Tabelle 141: CosT_8b0: *TERMINATE DF* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Terminierung

Tabelle 142: CosT_f24: *TERMINATE DF* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis CosH_3ff: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16080 - (N048.838) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das TERMINATE DF-Kommando zusätzliche Trailer verwendet.**[<=]**

14.2.8.3 Kommandoabarbeitung innerhalb der Karte

G2_N048.840.a - (N048.840)a K_COS
Das COS MUSS die TERMINATE DF-Variante aus
- 14.2.8.1- Use Case Terminieren eines Ordners
unterstützen.**[<=]**

A_16081 - (N048.840)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere TERMINATE DF-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N048.842 - (N048.842) K_COS
affectedObject MUSS gleich *currentFolder* gesetzt werden.**[<=]**

G2_N048.844 - (N048.844) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N048.852 - (N048.852) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert "Termination state" besitzt, dann MUSS als Trailer NoError verwendet werden.**[<=]**

G2_N048.856 - (N048.856) K_COS
Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert "Termination state" gesetzt werden.**[<=]**

A_16082 - (N048.860) K_TST

Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16083 - (N048.862) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N048.864 - (N048.864) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16084 - (N048.866) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_f24 ist herstellerspezifisch.
b. Jeder Trailer in CosT_f24 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.2.9 TERMINATE

Das Kommando TERMINATE überführt eine Datei, ein Schlüsselobjekt oder ein Passwortobjekt irreversibel in den Zustand "Termination state". Eine betroffene Datei wird vor der Operation ausgewählt. Dies geschieht vor dem Senden dieses TERMINATE-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*). Wenn ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, dann wird dieses durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachricht enthalten ist.

14.2.9.1 Use Case Terminieren einer Datei

In dieser Variante wird eine Datei terminiert und die APDU des TERMINATE-Kommandos enthält einen Parameter.

A_16085 - (N048.900) K_externeWelt {K_Karte}
Der Parameter *mode* MUSS anzeigen, dass *currentEF* zu terminieren ist.**[<=]**

A_16086 - (N048.903) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_847 verwendet werden.

Tabelle 143: CosT_847: *TERMINATE* aktuelle Datei

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E8'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	*mode*, der Wert '00' zeigt an, dass *currentEF* terminiert wird
P2	'00'	–

[<=]

14.2.9.2 Use Case Terminieren eines privaten oder symmetrischen Schlüsselobjektes

In dieser Variante wird ein privates Schlüsselobjekt oder ein symmetrisches Authentisierungsobjekt terminiert und die APDU des TERMINATE-Kommandos enthält zwei Parameter:

A_16087 - (N048.910) K_externeWelt {K_Karte}
Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu terminieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.**[<=]**

A_16088 - (N048.912) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

A_16089 - (N048.914) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_1cd verwendet werden.

Tabelle 144: CosT_1cd: *TERMINATE* privates oder symmetrisches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E8'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'20'	*mode*, hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	*reference* auf ein Schlüsselobjekt

[<=]

14.2.9.3 Use Case Terminieren eines öffentlichen Schlüsselobjektes

In dieser Variante wird ein öffentliches Schlüsselobjekt terminiert und die APDU des TERMINATE-Kommandos enthält zwei Parameter:

A_16090 - (N048.920) K_externeWelt {K_Karte}
Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu terminieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.**[<=]**

A_16091 - (N048.922) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt.**[<=]**

A_16092 - (N048.924) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_26d verwendet werden.

Tabelle 145: CosT_26d: *TERMINATE* öffentliches Schlüsselobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E8'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'21'	*mode*, hier: Schlüsselreferenz im Datenteil
P2	'00'	-
Data	'XX…XX'	'83 – I2OS(OctetLength(*reference*), 1) – *reference*'

[<=]

14.2.9.4 Use Case Terminieren eines Passwortobjektes

In dieser Variante wird ein Passwortobjekt terminiert und die APDU des TERMINATE-Kommandos enthält zwei Parameter:

A_16093 - (N048.930) K_externeWelt {K_Karte}
Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu terminieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.**[<=]**

A_16094 - (N048.932) K_externeWelt {K_Karte}
Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.**[<=]**

A_16095 - (N048.934) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_770 verwendet werden.

Tabelle 146: CosT_770: *TERMINATE* Passwortobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E8'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'10'	*mode*, hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	'XX'	*reference* auf ein Passwortobjekt

[<=]

14.2.9.5 Antwort der Karte auf Terminieren von Datei, Schlüssel- oder Passwortobjekt

Tabelle 147: CosT_722: *TERMINATE* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Terminierung

Tabelle 148: CosT_6af: *TERMINATE* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	VolatileKeyWithoutLCS	volatile Schlüssel vom Kommando nicht unterstützt
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 88'	KeyNotFound	Schlüsselobjekt nicht gefunden
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_8f6: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16096 - (N048.948) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das TERMINATE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.2.9.6 Kommandoabarbeitung innerhalb der Karte

G2_N048.950.a - (N048.950)a K_COS
Das COS MUSS die TERMINATE-Varianten aus
- 14.2.9.1- Use Case Terminieren einer Datei,
- 14.2.9.2- Use Case Terminieren eines privaten oder symmetrischen Schlüsselobjektes,
- 14.2.9.3- Use Case Terminieren eines öffentlichen Schlüsselobjektes und
- 14.2.9.4- Use Case Terminieren eines Passwortobjektes

unterstützen.**[<=]**

A_16097 - (N048.950)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere TERMINATE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N048.954.a.1 - (N048.954)a.1 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und*channelContext*. *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N048.954.a.2 - (N048.954)a.2 K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '00' besitzt und*channelContext*. *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N048.954.b - (N048.954)b K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '20' oder '21' besitzt, dann gilt
affectedObject = SearchKey(
channelContext. *currentFolder*,
reference,
"WildCard"
). Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht

erfolgreich ist. Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N048.954.c - (N048.954)c K_COS
Wenn der Parameter *mode* in der Kommandonachricht den Wert '10' besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N048.955 - (N048.955) K_COS
Wenn *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer VolatileKeyWithoutLCS terminieren.**[<=]**

G2_N048.957 - (N048.957) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N048.960 - (N048.960) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert "Termination state" besitzt, dann MUSS als Trailer NoError verwendet werden.**[<=]**

G2_N048.963 - (N048.963) K_COS
Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert "Termination state" gesetzt werden.**[<=]**

A_16098 - (N048.966) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16099 - (N048.969) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N048.972 - (N048.972) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16100 - (N048.975) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_6af ist herstellerspezifisch.
b. Jeder Trailer in CosT_6af MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.3 Zugriff auf Daten in transparenten EF

Es ist möglich auf Daten im *body* eines transparenten EF lesend (READ BINARY-Kommando) oder schreibend (UPDATE BINARY-Kommando) zuzugreifen. Zudem ist es möglich den Dateninhalt von *body* durch Nullen zu löschen (ERASE BINARY-Kommando). Sofern die maximale Dateigröße nicht erreicht ist, ist es möglich *body* um weitere Daten zu ergänzen (WRITE BINARY-Kommando). Die Größe des Teils von *body*, der für Lesezugriffe zugänglich ist, lässt sich auch verringern (SET LOGIGAL EOF-Kommando).

Die Kommandos in diesem Unterkapitel unterstützen zwei Varianten:

1. Variante ohne *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass für eine erfolgreiche Kommandoabarbeitung *currentEF* notwendigerweise ein transparentes EF ist. Die Variable *currentEF* lässt sich unter anderem durch gewisse Varianten des SELECT-Kommandos setzen.
2. Variante mit *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass das vom Kommando betroffene EF erst während der Kommandoabarbeitung gesetzt wird. Ein vorausgehendes SELECT-Kommando ist also nicht notwendig. In der Variante mit *shortFileIdentifier* lassen sich nur Dateien adressieren, die *currentFolder* zugeordnet sind.

14.3.1 ERASE BINARY

Das Kommando ERASE BINARY ersetzt bereits vorhandene Daten im *body* eines transparenten EF durch Oktette mit dem Wert '00'. Das betroffene transparente EF wird vor der Löschoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses ERASE BINARY-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses ERASE BINARY-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Informationen im *body* gelöscht werden, bestimmt der Offset, der als Parameter in der Kommandonachricht enthalten ist.

14.3.1.1 Use Case Löschen ohne shortFileIdentifier in transparenten EF

In dieser Variante werden Daten in einem transparenten EF gelöscht, ohne das Attribut *positionLogicalEndOfFile* zu verändern. Die APDU des ERASE BINARY-Kommandos enthält einen Parameter:

A_16101 - (N049.000) K_externeWelt {K_Karte}
Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = [`0000', '7FFF'] sein (vergleiche (N011.500)a).**[<=]**

A_16102 - (N049.100) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_238 verwendet werden.

Tabelle 149: CosT_238: *ERASE BINARY*, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'0E'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	(*offset* – P2) / 256, MSByte von *offset*
P2	'XX'	*offset* mod 256, LSByte von *offset*

[<=]

14.3.1.2 Use Case Löschen mit shortFileIdentifier in transparenten EF

In dieser Variante werden Daten in einem transparenten EF gelöscht, ohne das Attribut *positionLogicalEndOfFile* zu verändern. Die APDU des ERASE BINARY-Kommandos enthält zwei Parameter:

A_16103 - (N049.200) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16104 - (N049.300) K_externeWelt {K_Karte}
Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = ['00', 'FF'] sein.**[<=]**

A_16105 - (N049.400) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_3f3 verwendet werden.

Tabelle 150: CosT_3f3: *ERASE BINARY*, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'0E'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	128 + *shortFileIdentifier*, das heißt '80' + *shortFileIdentifier*
P2	'XX'	*offset*

[<=]

14.3.1.3 Antwort der Karte auf Löschen in transparenten EF

Tabelle 151: CosT_adb: *ERASE BINARY* Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Löschvorgang

Tabelle 152: CosT_4c8: *ERASE BINARY* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht transparent
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6B 00'	OffsetTooBig	Parameter *offset* in Kommando–APDU ist zu groß

Hinweis CosH_2f9: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16106 - (N049.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das ERASE BINARY-Kommando zusätzliche Trailer verwendet.**[<=]**

14.3.1.4 Kommandoabarbeitung innerhalb der Karte

G2_N049.600.a - (N049.600)a K_COS
Das COS MUSS die ERASE BINARY-Varianten aus
- 14.3.1.1- Use Case Löschen ohne shortFileIdentifier in transparenten EF und
- 14.3.1.2- Use Case Löschen mit shortFileIdentifier in transparenten EF

unterstützen.**[<=]**

A_16107 - (N049.600)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere ERASE BINARY-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16108 - (N049.700) a K_COS
Wenn die APDU des ERASE BINARY-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N049.700.a.1 - (N049.700)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N049.700.a.2 - (N049.700)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N049.700.b.1 - (N049.700)b.1 K_COS
Wenn die APDU des ERASE BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N049.700.b.2 - (N049.700)b.2 K_COS
Wenn die APDU des ERASE BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N049.800 - (N049.800) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N049.900 - (N049.900) K_COS
Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N050.000 - (N050.000) K_COS

Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.**[<=]**

G2_N050.100.a - (N050.100)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, genau dann MUSS *affectedObject.body* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen, sofern diese vorhanden ist.**[<=]**

A_16109 - (N050.100)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.body* mit oder ohne Transaktionsschutz geändert wird.**[<=]**

A_16110 - (N050.190) K_COS
Wenn *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.body*
a. DARF NICHT zum Abbruch des Kommandos führen.
b. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N050.200.a - (N050.200)a K_COS
Wenn *offset* kleiner als *affectedObject. positionLogicalEndOfFile* ist, dann MÜSSEN die durch *offset* gekennzeichnete Stelle und alle weiteren Oktette in *affectedObject.body* auf den Wert '00' gesetzt werden.**[<=]**

G2_N050.200.b - (N050.200)b K_COS
Wenn *offset* kleiner als *affectedObject. positionLogicalEndOfFile* ist und *affectedObject.body* ist durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.**[<=]**

A_16111 - (N050.300) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16112 - (N050.400) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N050.500 - (N050.500) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16113 - (N050.600) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_4c8 ist herstellerspezifisch.
b. Jeder Trailer in CosT_4c8 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N050.700) Diese Anforderung ist absichtlich leer.

(N050.800) Diese Anforderung ist absichtlich leer.

Hinweis CosH_3ae: Es ist aus funktionaler Sicht irrelevant, ob die in (N050.200)a beschriebene erase-Funktion an der Stelle affectedObject.positionLogicalEndOfFile *stoppt, oder bis zum Ende von*

affectedObject.body *weiterarbeitet, weil sich diese beiden Fälle an der Kartenschnittstelle nicht unterscheiden.*

14.3.2 READ BINARY

Das Kommando READ BINARY dient dem Auslesen von Informationen aus dem *body* eines transparenten EF. Deshalb enthält das Datenfeld der Antwortnachricht (Teile von) *body*. Das betroffene transparente EF wird vor der Leseoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses READ BINARY-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses READ BINARY-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Informationen aus *body* ausgelesen werden, bestimmen Offset und Länge, die als Parameter in der Kommandonachricht enthalten sind.

14.3.2.1 Use Case Lesen ohne shortFileIdentifier in transparenten EF

In dieser Variante enthält die APDU des READ BINARY-Kommandos zwei Parameter:

A_16116 - (N050.900) K_externeWelt {K_Karte}
Der Parameter *offset* bestimmt, ab welcher Position gelesen wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = ['0000', '7FFF'] sein (vergleiche (N011.500)a).**[<=]**

A_16119 - (N051.000) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16120 - (N051.100) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_a50 verwendet werden.

Tabelle 153: CosT_a50: *READ BINARY* ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'B0'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	(*offset* – P2) / 256, MSByte von *offset*
P2	'XX'	*offset* mod 256, LSByte von *offset*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.3.2.2 Use Case Lesen mit shortFileIdentifier in transparenten EF

In dieser Variante enthält die APDU des READ BINARY-Kommandos drei Parameter:

A_16121 - (N051.200) K_externeWelt {K_Karte}

Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16122 - (N051.300) K_externeWelt {K_Karte}
Der Parameter *offset* bestimmt, ab welcher Position gelesen wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = ['00', 'FF'] sein.**[<=]**

A_16123 - (N051.400) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16124 - (N051.500) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_301 verwendet werden.

Tabelle 154: CosT_301: *READ BINARY* mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'B0'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	128 + *shortFileIdentifier*, das heißt '80' + *shortFileIdentifier*
P2	'XX'	*offset*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.3.2.3 Antwort der Karte auf Lesen in transparenten EF

Tabelle 155: CosT_1a8: *READ BINARY* Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'		Ausgelesene Daten
Trailer	Inhalt	Beschreibung
'62 81'	CorruptDataWarning	Möglicherweise sind die Antwortdaten korrupt
'62 82'	EndOfFileWarning	Weniger Daten vorhanden, als mittels Ne angefordert
'90 00'	NoError	Erfolgreiche Leseoperation

Tabelle 156: CosT_1b0: *READ BINARY* Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 81'	WrongFileType	Ausgewähltes EF ist nicht transparent
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6B 00'	OffsetTooBig	Parameter *offset* in Kommando-APDU ist zu groß

Hinweis CosH_782: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16125 - (N051.600) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das READ BINARY-Kommando zusätzliche Trailer verwendet.**[<=]**

14.3.2.4 Kommandoabarbeitung innerhalb der Karte

G2_N051.700.a - (N051.700).a K_COS
Das COS MUSS die READ BINARY-Varianten aus
- 14.3.2.1- Use Case Lesen ohne shortFileIdentifier in transparenten EF und
- 14.3.2.2- Use Case Lesen mit shortFileIdentifier in transparenten EF

unterstützen.**[<=]**

A_16126 - (N051.700)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere READ BINARY-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16127 - (N051.800)a K_COS
Wenn die APDU des READ BINARY-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N051.800.a(.1) - (N051.800)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N051.800.a.2 - (N051.800)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N051.800.b.1 - (N051.800)b.1 K_COS
Wenn die APDU des READ BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N051.800.b.2 - (N051.800)b.2 k_COS
Wenn die APDU des READ BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N051.900 - (N051.900) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N052.000 - (N052.000) K_COS
Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N052.100 - (N052.100) K_COS
Wenn *offset* größer oder gleich *affectedObject.positionLogicalEndOfFile* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.**[<=]**

A_16128 - (N052.200) K_COS
Wenn *affectedObject.flagChecksum* den Wert True hat und die Daten von *affectedObject.body* inkonsistent zur Checksumme sind, genau dann MUSS eine der beiden folgenden Verhaltensweisen implementiert sein:**[<=]**

G2_N052.200.a - (N052.200)a K_COS
Möglichkeit 1: Als Trailer MUSS CorruptDataWarning gewählt werden.**[<=]**

A_16129 - (N052.200)b K_COS
Möglichkeit 2: Die Kommandobearbeitung MUSS gemäß (N031.940) stoppen.**[<=]**

G2_N052.300 - (N052.300) K_COS
Wenn das LeFeld der Kommando-APDU keine WildCard enthält und (*offset* + *length*) größer als *affectedObject.positionLogicalEndOfFile* ist, genau dann MUSS als Trailer EndOfFileWarning gewählt werden.**[<=]**

G2_N052.400 - (N052.400) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

G2_N052.500.a - (N052.500)a K_COS
Aus dem Oktettstring von *affectedObject.body* MÜSSEN die durch *offset* gekennzeichnete Position und die nachfolgenden Oktette übernommen werden.**[<=]**

A_16130 - (N052.500)b K_COS
Es DÜRFEN NICHT mehr Oktette übernommen werden, als durch Ne angegeben.**[<=]**

G2_N052.500.c - (N052.500)c K_COS
Die Übernahme der Oktette MUSS so gestoppt werden, dass weder das Oktett an der Position *affectedObject.positionLogicalEndOfFile* noch nachfolgende Oktette übernommen werden.**[<=]**

A_16131 - (N052.600)a K_TST
Die Priorität der Trailer in CosT_1b0 MUSS herstellerspezifisch sein.**[<=]**

G2_N052.600.b - (N052.600)b K_COS
Jeder Trailer in CosT_1b0 MUSS eine höhere Priorität als CorruptDataWarning haben.**[<=]**

G2_N052.600.c - (N052.600)c K_COS
CorruptDataWarning MUSS eine höhere Priorität als EndOfFileWarning haben.**[<=]**

G2_N052.600.d - (N052.600)d K_COS
EndOfFileWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N052.700) Diese Anforderung ist absichtlich leer.

(N052.800) Diese Anforderung ist absichtlich leer.

14.3.3 SEARCH BINARY

A_16132 - (N052.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das SEARCH BINARY-Kommando gemäß [ISO/IEC 7816-4]
a. unterstützt oder
b. ablehnt.**[<=]**

14.3.4 SET LOGICAL EOF

Hinweis CosH_756: Dieses Kommando ist nicht in der Normenreihe ISO/IEC 7816 enthalten. Wegen ähnlicher Funktionalität teilt sich dieses Kommando das INS Byte mit ERASE BINARY.

Das Kommando SET LOGICAL EOF verändert den Wert des Attributes *positionLogicalEndOfFile* eines transparenten EF, wodurch Daten gelöscht werden. Das betroffene transparente EF wird vor der Löschoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses SET LOGICAL EOF-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses SET LOGICAL EOF-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Auf welchen Wert das Attribut *positionLogicalEndOfFile* gesetzt wird und damit welche Daten in *body* gelöscht werden, bestimmt der Offset, der als Parameter in der Kommandonachricht enthalten ist.

14.3.4.1 Use Case Setzen logical EOF ohne shortFileIdentifier

In dieser Variante werden Daten in einem transparenten EF gelöscht und das Attribut *positionLogicalEndOfFile* wird verändert.

A_16133 - (N052.930) K_externeWelt {K_Karte}
Die APDU des SET LOGICAL EOF-Kommandos enthält einen Parameter:
a. Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = [`0000', '7FFF'] sein (vergleiche (N011.500)a).**[<=]**

A_16134 - (N052.932) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_be2 verwendet werden.

Tabelle 157: CosT_be2: SET LOGICAL EOF ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'0E'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	(*offset* – P2) / 256, MSByte von *offset*
P2	'XX'	*offset* mod 256, LSByte von *offset*

[<=]

14.3.4.2 Use Case Setzen logical EOF mit shortFileIdentifier

In dieser Variante werden Daten in einem transparenten EF gelöscht und das Attribut *positionLogicalEndOfFile* wird verändert.

A_16135 - (N052.934) K_externeWelt {K_Karte}
Die APDU des SET LOGICAL EOF-Kommandos enthält zwei Parameter:

a. Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.

b. Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = ['00', 'FF'] sein.

[<=]

A_16136 - (N052.936) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_164 verwendet werden.

Tabelle 158: CosT_164: SET LOGICAL EOF mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'0E'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	128 + *shortFileIdentifier*, das heißt '80' + *shortFileIdentifier*
P2	'XX'	*offset*

[<=]

14.3.4.3 Antwort der Karte auf Setzen logical EOF in transparenten EF

Tabelle 159: CosT_77c: SET LOGICAL EOF Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Löschvorgang

Tabelle 160: CosT_a26: SET LOGICAL EOF Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht transparent
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6B 00'	OffsetTooBig	Parameter *offset* in Kommando–APDU ist zu groß

Hinweis CosH_a51: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16137 - (N052.938) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das SET LOGICAL EOF-Kommando zusätzliche Trailer verwendet.**[<=]**

14.3.4.4 Kommandoabarbeitung innerhalb der Karte

G2_N052.940.a - (N052.940)a K_COS
Das COS MUSS die SET LOGICAL EOF-Varianten aus
- 14.3.4.1- Use Case Setzen logical EOF ohne shortFileIdentifier und
- 14.3.4.2- Use Case Setzen logical EOF mit shortFileIdentifier

unterstützen.**[<=]**

A_16138 - (N052.940)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere SET LOGICAL EOF-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16139 - (N053.942)a K_COS
Wenn die APDU des SET LOGICAL EOF-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N052.942.a.1 - (N052.942)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N052.942.a.2 - (N052.942)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N052.942.b.1 - (N052.942)b.1 K_COS
Wenn die APDU des SET LOGICAL EOF-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N052.942.b.2 - (N052.942)b.2 K_COS

Wenn die APDU des SET LOGICAL EOF-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N052.944 - (N052.944) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N052.946 - (N052.946) K_COS
Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N052.948 - (N052.948) K_COS
Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.**[<=]**

G2_N052.950.a - (N052.950)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, genau dann MUSS *affectedObject.body* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen, sofern diese vorhanden ist.**[<=]**

A_16140 - (N052.950)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.body* mit oder ohne Transaktionsschutz geändert wird.**[<=]**

A_16141 - (N052.952) K_COS
Wenn *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.body*
a. DARF NICHT zum Abbruch des Kommandos führen.
b. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N052.954.a.1 - (N052.954)a.1 K_COS
Wenn *offset* kleiner als *affectedObject.positionLogicalEndOfFile* ist, dann MUSS, falls *affectedObject.body* durch eine Checksumme geschützt ist, diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.**[<=]**

G2_N052.954.a.2 - (N052.954)a.2 K_COS
Wenn *offset* kleiner als *affectedObject.positionLogicalEndOfFile* ist, dann MUSS *affectedObject.positionLogicalEndOfFile* auf den Wert von *offset* geändert werden. **[<=]**

Hinweis CosH_d76: Aus Sicherheitsgründen erscheint es sinnvoll, die durch offset *gekennzeichnete Stelle und alle weiteren Oktette in* affectedObject.body *auf den Wert '00' zu setzten. Ob so eine Aktion erfolgt, ist mit den normativen Mitteln aus diesem Dokument nicht testbar, vergleiche auch (N054.500)a und (N055.258).*

G2_N052.954.b - (N052.954)b K_COS
Wenn *offset* größer gleich als *affectedObject.positionLogicalEndOfFile* ist, dann DARF *affectedObject.positionLogicalEndOfFile* NICHT geändert werden.**[<=]**

A_16142 - (N052.956) K_TST

Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16143 - (N052.958) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N052.960 - (N052.960) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16144 - (N052.962) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_a26 ist herstellerspezifisch.
b. Jeder Trailer in CosT_a26 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

Hinweis CosH_c51: Es ist aus funktionaler Sicht irrelevant, ob die in CosH_d76 beschriebene erase-Funktion an der Stelle positionLogicalEndOfFile *stoppt, oder bis zum Ende von* affectedObject.body *weiterarbeitet, weil sich diese beiden Fälle an der Kartenschnittstelle nicht unterscheiden.*

14.3.5 UPDATE BINARY

Das Kommando UPDATE BINARY ersetzt Daten im *body* eines transparenten EF durch Daten, die im Datenfeld der Kommandonachricht enthalten sind. Das betroffene transparente EF wird vor der Schreiboperation ausgewählt. Dies geschieht entweder vor dem Senden dieses UPDATE BINARY-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses UPDATE BINARY-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Informationen im *body* geändert werden, bestimmen Offset und Schreibdaten, die als Parameter in der Kommandonachricht enthalten sind.

14.3.5.1 Use Case Schreiben ohne shortFileIdentifier in transparenten EF

In dieser Variante enthält die APDU des UPDATE BINARY-Kommandos zwei Parameter:

A_16145 - (N053.000) K_externeWelt {K_Karte}
Der Parameter *offset* bestimmt, ab welcher Position geschrieben wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 32767] = [`0000', '7FFF'] sein (vergleiche (N011.500)a).**[<=]**

A_16146 - (N053.100) K_externeWelt {K_Karte}
Der Parameter *newData* enthält die neuen Daten, welche die, ab der durch *offset* gekennzeichneten Stelle, enthaltenen Daten in *body* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.**[<=]**

A_16147 - (N053.200) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_cdb verwendet werden.

Tabelle 161: CosT_cdb: UPDATE BINARY ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'D6'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	(*offset* – P2) / 256, MSByte von *offset*
P2	'XX'	*offset* mod 256, LSByte von *offset*
Data	'XX…XX'	*newData*

[<=]

14.3.5.2 Use Case Schreiben mit shortFileIdentifier in transparenten EF

In dieser Variante enthält die APDU des UPDATE BINARY-Kommandos drei Parameter:

A_16148 - (N053.300) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16149 - (N053.400) K_externeWelt {K_Karte}
Der Parameter *offset* bestimmt, ab welcher Position geschrieben wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = ['00', 'FF'] sein.**[<=]**

A_16150 - (N053.500) K_externeWelt {K_Karte}
Der Parameter *newData* enthält die neuen Daten, welche die ab der durch *offset* gekennzeichneten Stelle enthaltenen Daten in *body* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.**[<=]**

A_16151 - (N053.600) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_e25 verwendet werden.

Tabelle 162: CosT_e25: UPDATE BINARY mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'D6'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	128 + *shortFileIdentifier*, das heißt '80' + *shortFileIdentifier*
P2	'XX'	*offset*
Data	'XX…XX'	*newData*

[<=]

14.3.5.3 Antwort der Karte auf Schreiben in transparenten EF

Tabelle 163: CosT_94b: UPDATE BINARY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Schreibvorgang

Tabelle 164: CosT_c75: UPDATE BINARY Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht transparent
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 84'	DataTooBig	Parameter *newData* ragt über das Dateiende hinaus
'6B 00'	OffsetTooBig	Parameter *offset* in Kommando–APDU ist zu groß

Hinweis CosH_71e: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16152 - (N053.700) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das UPDATE BINARY-Kommando zusätzliche Trailer verwendet.**[<=]**

14.3.5.4 Kommandoabarbeitung innerhalb der Karte

G2_N053.800.a - (N053.800)a K_COS
Das COS MUSS die UPDATE BINARY-Varianten aus
- 14.3.5.1- Use Case Schreiben ohne shortFileIdentifier in transparenten EF und
- 14.3.5.2- Use Case Schreiben mit shortFileIdentifier in transparenten EF

unterstützen.**[<=]**

A_16153 - (N053.800)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere UPDATE BINARY-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16154 - (N053.900)a K_COS
Wenn die APDU des UPDATE BINARY-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N053.900.a.1 - (N053.900)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N053.900.a.2 - (N053.900)a.2 K_COS

Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N053.900.b.1 - (N053.900)b.1 K_COS
Wenn die APDU des UPDATE BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N053.900.b.2 - (N053.900)b.2 K_COS
Wenn die APDU des UPDATE BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N054.000 - (N054.000) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N054.100 - (N054.100) K_COS
Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N054.200 - (N054.200) K_COS
Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.**[<=]**

G2_N054.300 - (N054.300) K_COS
Wenn (*offset* + OctetLength(*newData*)) größer als *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer DataTooBig terminieren.**[<=]**

G2_N054.400.a - (N054.400)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, genau dann MUSS *affectedObject.body* und gegebenenfalls *affectedObject.positionLogicalEndOfFile* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen, sofern diese vorhanden ist.**[<=]**

A_16155 - (N054.400)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.body* mit oder ohne Transaktionsschutz geändert wird.**[<=]**

A_16156 - (N054.490) K_COS
Wenn *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.body*
a. DARF NICHT zum Abbruch des Kommandos führen.
b. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N054.500.a - (N054.500)a K_COS
Wenn *affectedObject.positionLogicalEndOfFile* kleiner als *offset* ist, dann MUSS das Oktett an der Position *affectedObject.positionLogicalEndOfFile* und alle nachfolgenden Oktette bis zur Position (*offset* – 1) auf den Wert '00' gesetzt werden.**[<=]**

G2_N054.500.b - (N054.500)b K_COS

Die durch *offset* gekennzeichnete Stelle in *affectedObject.body* und die folgenden Oktette MÜSSEN durch *newData* ersetzt werden.**[<=]**

G2_N054.500.c - (N054.500)c K_COS
Wenn *positionLogicalEndOfFile* kleiner als (*offset* + OctetLength(*newData*)) ist, dann MUSS *positionLogicalEndOfFile* = (*offset* + OctetLength(*newData*)) gesetzt werden.**[<=]**

G2_N054.500.d - (N054.500)d K_COS
Wenn *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.**[<=]**

A_16157 - (N054.600) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16158 - (N054.700) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N054.800 - (N054.800) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16159 - (N054.900) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_c75 ist herstellerspezifisch.
b. Jeder Trailer in CosT_c75 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N055.000) Diese Anforderung ist absichtlich leer.

(N055.100) Diese Anforderung ist absichtlich leer.

14.3.6 WRITE BINARY

Das Kommando WRITE BINARY fügt den vorhandenen Daten im *body* eines transparenten EF Daten hinzu, die im Datenfeld der Kommandonachricht enthalten sind. Das betroffene transparente EF wird vor der Schreiboperation ausgewählt. Dies geschieht entweder vor dem Senden dieses WRITE BINARY-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses WRITE BINARY-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Wieviele Daten *body* hinzugefügt werden, bestimmen Schreibdaten, die als Parameter in der Kommandonachricht enthalten sind.

14.3.6.1 Use Case Anfügen ohne shortFileIdentifier in transparenten EF

In dieser Variante enthält die APDU des WRITE BINARY-Kommandos einen Parameter:

A_16160 - (N055.200) K_externeWelt {K_Karte}
Der Parameter *newData* enthält die neuen Daten, welche die, ab der durch *positionLogicalEndOfFile* gekennzeichneten Stelle, enthaltenen Daten in *body* ergänzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.**[<=]**

A_16161 - (N055.205) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_17f verwendet werden.

Tabelle 165: CosT_17f: WRITE BINARY ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'D0'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Bit b8 = 0 ➔ die Kommando APDU enthält keinen *shortFileIdentifier*
P2	'00'	fester Wert
Data	'XX…XX'	*newData*

[<=]

14.3.6.2 Use Case Anfügen mit shortFileIdentifier in transparenten EF

In dieser Variante enthält die APDU des WRITE BINARY-Kommandos zwei Parameter:

A_16162 - (N055.220) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16163 - (N055.223) K_externeWelt {K_Karte}
Der Parameter *newData* enthält die neuen Daten, welche die, ab der durch *positionLogicalEndOfFile* gekennzeichneten Stelle, enthaltenen Daten in *body* ergänzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.**[<=]**

A_16164 - (N055.226) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_415 verwendet werden.

Tabelle 166: CosT_415: WRITE BINARY mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'D0'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	128 + *shortFileIdentifier*, das heißt '80' + *shortFileIdentifier*
P2	'00'	fester Wert
Data	'XX…XX'	*newData*

[<=]

14.3.6.3 Antwort der Karte auf Anfügen in transparenten EF

Tabelle 167: CosT_e5e: WRITE BINARY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Schreibvorgang

Tabelle 168: CosT_507: WRITE BINARY Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht transparent
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 84'	DataTooBig	Parameter *newData* ragt über das Dateiende hinaus

Hinweis CosH_cc5: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16165 - (N055.240) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das WRITE BINARY-Kommando zusätzliche Trailer verwendet.**[<=]**

14.3.6.4 Kommandoabarbeitung innerhalb der Karte

G2_N055.244.a - (N055.244)a K_COS
Das COS MUSS die WRITE BINARY-Varianten aus
- 14.3.6.1- Use Case Anfügen ohne shortFileIdentifier in transparenten EF und
- 14.3.6.2- Use Case Anfügen mit shortFileIdentifier in transparenten EF

unterstützen.**[<=]**

A_16166 - (N055.224)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere WRITE BINARY-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16167 - (N055.246)a K_COS
Wenn die APDU des WRITE BINARY-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N055.246.a.1 - (N055.246)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N055.246.a.2 - (N055.246)a.2 K_COS

Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N055.246.b.1 - (N055.246)b.1 K_COS
Wenn die APDU des WRITE BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N055.246.b.2 - (N055.246)b.2 K_COS
Wenn die APDU des WRITE BINARY-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N055.248 - (N055.248) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N055.250 - (N055.250) K_COS
Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N055.252 - (N055.252) K_COS
Wenn (*positionLogicalEndOfFile* + OctetLength(*newData*)) größer als *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer DataTooBig terminieren.**[<=]**

G2_N055.254.a - (N055.254)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, genau dann MÜSSEN *affectedObject.body* und *affectedObject.positionLogicalEndOfFile* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen, sofern diese vorhanden ist.**[<=]**

A_16168 - (N055.254)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.body* mit oder ohne Transaktionsschutz geändert wird.**[<=]**

A_16169 - (N055.256) K_COS
Wenn *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.body*
a. DARF NICHT zum Abbruch des Kommandos führen.
b. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N055.258.a - (N055.258)a K_COS
Die durch *positionLogicalEndOfFile* gekennzeichnete Stelle in *affectedObject.body* und die folgenden Oktette MÜSSEN durch *newData* ersetzt werden.**[<=]**

G2_N055.258.b - (N055.258)b K_COS
Das Attribut *positionLogicalEndOfFile* MUSS um OctetLength(*newData*) inkrementiert werden.**[<=]**

A_16170 - (N055.258)c K_COS

Wenn *affectedObject.body* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.body* ist.**[<=]**

A_16171 - (N055.260) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16172 - (N055.262) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N055.264 - (N055.264) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16173 - (N055.266) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_507 ist herstellerspezifisch.
b. Jeder Trailer in CosT_507 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N055.268) Diese Anforderung ist absichtlich leer.

(N055.270) Diese Anforderung ist absichtlich leer.

14.4 Zugriff auf strukturierte Daten

Es ist möglich auf Listenelemente von *recordList* in strukturierten EF lesend (READ RECORD) oder schreibend (UPDATE RECORD) zuzugreifen. Zudem lassen sich neue Listenelemente anlegen (APPEND RECORD). Es ist möglich, ein Listenelement (DELETE RECORD) oder dessen **Inhalt** zu löschen (ERASE RECORD). Listenelemente lassen sich aktivieren (ACTIVATE RECORD) und deaktivieren (DEACTIVATE RECORD), was sich auf die Nutzung des Rekordinhaltes auswirkt. Des Weiteren ist es möglich, in der Liste nach Elementen zu suchen, deren Inhalt zu einem frei wählbaren Suchmuster passt (SEARCH RECORD).

Die Kommandos in diesem Unterkapitel unterstützen zwei Varianten:

1. Variante ohne *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass für eine erfolgreiche Kommandoabarbeitung *currentEF* notwendigerweise ein strukturiertes EF ist. Die Variable *currentEF* lässt sich unter anderem durch gewisse Varianten des SELECT-Kommandos setzen.
2. Variante mit *shortFileIdentifier*: Diese Variante ist dadurch gekennzeichnet, dass das vom Kommando betroffene EF erst während der Kommandoabarbeitung gesetzt wird. Ein vorausgehendes SELECT-Kommando ist also nicht notwendig. In der Variante mit *shortFileIdentifier* lassen sich keine Dateien adressieren, die nicht *currentFolder* zugeordnet sind.

14.4.1 ACTIVATE RECORD

Das Kommando ACTIVATE RECORD aktiviert ein oder mehrere Listenelemente aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses ACTIVATE RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses ACTIVATE RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Listenelemente aktiviert werden, bestimmen Rekordnummer und Modus, welche als Parameter in der Kommandonachricht enthalten sind.

14.4.1.1 Use Case Aktivieren eines Rekords ohne shortFileIdentifier

In dieser Variante enthält die APDU des ACTIVATE RECORD-Kommandos zwei Parameter:

A_16223 - (N055.300) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16224 - (N055.400) K_externeWelt {K_Karte}
Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.**[<=]**

A_16225 - (N055.500) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_498 verwendet werden.

Tabelle 169: CosT_498: ACTIVATE RECORD, ein Rekord, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'08'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'04'	*mode*, Codierung '04' bedeutet "nutze Listenelement P1"

[<=]

14.4.1.2 Use Case Aktivieren eines Rekords mit shortFileIdentifier

In dieser Variante enthält die APDU des ACTIVATE RECORD-Kommandos drei Parameter:

A_16226 - (N055.600) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16227 - (N055.700) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16228 - (N055.800) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.**[<=]**

A_16229 - (N055.900) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_5ce verwendet werden.

Tabelle 170: CosT_5ce: ACTIVATE RECORD, ein Rekord, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'08'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '04' Codierung '04' bedeutet "nutze Listenelement P1"

[<=]

14.4.1.3 Use Case Aktivieren aller Rekords ab P1 ohne shortFileIdentifier

In dieser Variante enthält die APDU des ACTIVATE RECORD-Kommandos zwei Parameter:

A_16230 - (N056.000) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16231 - (N056.100) K_externeWelt {K_Karte}
Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.**[<=]**

A_16232 - (N056.200) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_ff2 verwendet werden.

Tabelle 171: CosT_ff2: ACTIVATE RECORD, alle Rekords ab P1, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'08'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'05'	*mode*, Codierung '05' bedeutet "nutze Listenelemente ab P1"

[<=]

14.4.1.4 Use Case Aktivieren aller Rekords ab P1 mit shortFileIdentifier

In dieser Variante enthält die APDU des ACTIVATE RECORD-Kommandos drei Parameter:

A_16233 - (N056.300) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16234 - (N056.400) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16235 - (N056.500) K_externeWelt {K_Karte}
Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.**[<=]**

A_16236 - (N056.600) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_7b6 verwendet werden.

Tabelle 172: CosT_7b6: ACTIVATE RECORD, alle Rekords ab P1, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'08'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '05' Codierung '05' bedeutet "nutze Listenelemente ab P1"

[<=]

14.4.1.5 Antwort der Karte auf Aktivieren von Rekords

Tabelle 173: CosT_066: ACTIVATE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Aktivierung

Tabelle 174: CosT_180: ACTIVATE RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoRecordLifeCycleStatus	Rekords in ausgewähltem EF besitzen keinen LCS
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement *recordNumber* existiert nicht

Hinweis CosH_c79: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16237 - (N056.700) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das ACTIVATE RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.1.6 Kommandoabarbeitung innerhalb der Karte

G2_N056.800.a - (N056.800)a K_COS
Das COS MUSS die ACTIVATE RECORD-Varianten aus
- 14.4.1.1- Use Case Aktivieren eines Rekords ohne shortFileIdentifier,
- 14.4.1.2- Use Case Aktivieren eines Rekords mit shortFileIdentifier,
- 14.4.1.3- Use Case Aktivieren aller Rekords ab P1 ohne shortFileIdentifier und
- 14.4.1.4- Use Case Aktivieren aller Rekords ab P1 mit shortFileIdentifier

unterstützen.**[<=]**

A_16238 - (N056.800)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere ACTIVATE RECORD-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16239 - (N056.900)a K_COS
Wenn die APDU des ACTIVATE RECORD-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N056.900.a.1 - (N056.900)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N056.900.a.2 - (N056.900)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N056.900.b.1 - (N056.900)b.1 K_COS
Wenn die APDU des ACTIVATE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N056.900.b.2 - (N056.900)b.2 K_COS
Wenn die APDU des ACTIVATE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N057.000 - (N057.000) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N057.100 - (N057.100) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N057.200 - (N057.200) K_COS
Wenn *affectedObject.flagRecordLifeCycleStatus* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer NoRecordLifeCycleStatus terminieren.**[<=]**

G2_N057.300 - (N057.300) K_COS
Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.**[<=]**

G2_N057.400.a - (N057.400)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, genau dann MUSS der *lifecycleStatus* mit Transaktionsschutz geändert werden.**[<=]**

A_16240 - (N057.400)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.body* mit oder ohne Transaktionsschutz geändert wird.**[<=]**

G2_N057.500 - (N057.500) K_COS
Wenn *mode* = '04' ist und der physikalische Wert von *lifeCycleStatus* des durch *recordNumber* adressierten *record* in *affectedObject.recordList* bereits den Wert "Operational state (active)" besitzt, dann MUSS als Trailer NoError verwendet werden.**[<=]**

A_16241 - (N057.600) K_COS
Die physikalischen Werte von *lifeCycleStatus* der durch *recordNumber* und *mode* adressierten *record* in *affectedObject.recordList* MÜSSEN auf den Wert "Operational state (active)" gesetzt werden.**[<=]**

G2_N057.600.a - (N057.600)a K_COS
Wenn *mode* den Wert '04' besitzt, dann DARF NUR nur das durch*recordNumber* adressierte Listenelement betroffen sein.**[<=]**

G2_N057.600.b - (N057.600)b K_COS
Wenn *mode* den Wert '05' besitzt, dann MUSS das durch*recordNumber* adressierte Listenelement und alle folgenden betroffen sein.**[<=]**

A_16242 - (N057.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16244 - (N058.000)a K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
1. entweder als Trailer MemoryFailure verwendet werden,
2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N058.000.b - (N058.000)b K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16245 - (N058.000)c K_COS
Für die Priorität der Trailer gilt:
1. Die Priorität der Trailer in CosT_180 ist herstellerspezifisch.
2. Jeder Trailer in CosT_180 MUSS eine höhere Priorität als UpdateRetryWarning haben.
3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N058.100) Diese Anforderung ist absichtlich leer.

(N058.200) Diese Anforderung ist absichtlich leer.

14.4.2 APPEND RECORD

Das Kommando APPEND RECORD fügt ein neues Listenelement an *recordList* eines strukturierten EF an, wobei die Daten für den Oktettstring des neuen Listenelementes im Datenfeld der Kommandonachricht enthalten sind. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses APPEND RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses APPEND RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde.

14.4.2.1 Use Case Anlegen neuer Rekord, ohne shortFileIdentifier

Die APDU des APPEND RECORD-Kommandos enthält einen Parameter:

A_16247 - (N058.300) K_externeWelt {K_Karte}
Der Parameter *recordData* enthält die Daten des neuen Rekords. Der Parameter *recordData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *recordData* MUSS aus dem in (N007.700)a definierten Bereich gewählt werden.**[<=]**

A_16248 - (N058.400) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_74c verwendet werden.

Tabelle 175: CosT_74c: APPEND RECORD, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E2'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Parameter ohne Bedeutung
P2	'00'	Parameter ohne Bedeutung
Data	'XX…XX'	*recordData*

[<=]

14.4.2.2 Use Case Anlegen neuer Rekords, mit shortFileIdentifier

Die APDU des APPEND RECORD-Kommandos enthält zwei Parameter:

A_16249 - (N058.500) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16250 - (N058.600) K_externeWelt {K_Karte}
Der Parameter *recordData* enthält die Daten des neuen Rekords. Der Parameter *recordData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *recordData* MUSS aus dem in (N007.700)a definierten Bereich gewählt werden.**[<=]**

A_16251 - (N058.700) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_057 verwendet werden.

Tabelle 176: CosT_057: APPEND RECORD, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E2'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Parameter ohne Bedeutung
P2	'XX'	8 *shortFileIdentifier*, das heißt *shortFileIdentifier* << 3
Data	'XX…XX'	*recordData*

[<=]

14.4.2.3 Antwort der Karte auf Anlegen eines neuen Rekords

Tabelle 177: CosT_71a: APPEND RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Hinzufügen eines Rekords

Tabelle 178: CosT_8cd: APPEND RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'67 00'	WrongRecordLength	*recordData* hat nicht die richtige Länge
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 84'	FullRecordList	Rekordliste lässt keine weiteren Elemente zu
'6A 84'	OutOfMemory	Zu viele Oktette in *recordData*

Hinweis CosH_5a5: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16252 - (N058.800) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das APPEND RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.2.4 Kommandoabarbeitung innerhalb der Karte

G2_N058.900.a - (N058.900)a K_COS
Das COS MUSS die APPEND RECORD-Varianten aus
- 14.4.2.1- Use Case Anlegen neuer Rekord, ohne shortFileIdentifier und
- 14.4.2.2- Use Case Anlegen neuer Rekords, mit shortFileIdentifier

unterstützen.**[<=]**

A_16253 - (N058.900)b K_TST
Die funktionale Eignung DARF einem Prüfling NICHT mit der Begründung verwehrt werden weitere APPEND RECORD-Varianten würden
1. unterstützt oder
2. abgelehnt.**[<=]**

A_16254 - (N059.000)a K_COS
Wenn die APDU des APPEND RECORD-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N059.000.a.1 - (N059.000)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N059.000.a.2 - (N059.000)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N059.000.b.1 - (N059.000)b.1 K_COS
Wenn die APDU des APPEND RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N059.000.b.2 - (N059.000)b.2 K_COS
Wenn die APDU des APPEND RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N059.100 - (N059.100) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N059.200 - (N059.200) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

(N059.300) Diese Anforderung ist absichtlich leer.

G2_N059.400 - (N059.400) K_COS
Wenn die Anzahl der Listenelemente in *affectedObject.recordList* gleich *affectedObject.maximumNumberOfRecords* ist und *affectedObject* vom Typ linear fixes EF oder linear variables EF ist, genau dann MUSS das Kommando mit dem Trailer FullRecordList terminieren.**[<=]**

(N059.500) Diese Anforderung ist absichtlich leer.

(N059.600) Diese Anforderung ist absichtlich leer.

A_16255 - (N059.650) K_COS
Das COS MUSS folgende Schritte ausführen:**[<=]**

A_16256 - (N059.650)a K_COS
Schritt 1: Das COS MUSS die Anzahl der Oktette in *recordData* mit *affectedObject.maximumRecordLength* vergleichen.**[<=]**

G2_N059.650.a.1 - (N059.650)a.1 K_COS
Wenn *affectedObject* vom Typ linear fixes EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.**[<=]**

G2_N059.650.a.2 - (N059.650)a.2 K_COS
Wenn *affectedObject* vom Typ zyklisches EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.**[<=]**

G2_N059.650.a.3.i - (N059.650)a.3.i
Wenn *affectedObject* vom Typ linear variables EF ist und die Anzahl Oktette in *recordData* größer als *affectedObject.maximumRecordLength* ist, dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.**[<=]**

G2_N059.650.a.3.ii - (N059.650)a.3.ii K_COS
Wenn *affectedObject* vom Typ linear variables EF ist die Anzahl Oktette in den Oktettstrings aller *record* von *recordList* nach durchgeführter Listenerweiterung größer als *affectedObject.numberOfOctet* wäre, genau dann MUSS das Kommando mit dem Trailer OutOfMemory terminieren.**[<=]**

A_16257 - (N059.650)b K_COS
Schritt 2: Das COS MUSS die Art des Transaktionsschutzes festlegen.**[<=]**

G2_N059.650.b.1 - (N059.650)b.1 K_COS

Wenn *affectedObject.flagTransactionMode* den Wert True hat, dann MUSS *affectedObject.recordList* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.recordList* oder der Checksumme des neu angelegten Rekords umfassen, sofern diese vorhanden sind.**[<=]**

A_16258 - (N059.650)b.2 K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.recordList* mit oder ohne Transaktionsschutz geändert wird.**[<=]**

A_16259 - (N059.650)c K_COS
Schritt 3: Wenn *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.recordList*
1. DARF NICHT zum Abbruch des Kommandos führen.
2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

A_16263 - (N059.650)d K_COS
Schritt 4: Das COS MUSS *affectedObject.recordList* wie folgt einen neuen Eintrag hinzufügen:**[<=]**

G2_N059.650.d.1 - (N059.650)d.1 K_COS
Wenn *affectedObject* vom Typ linear fixes EF oder linear variables EF ist, dann wird ein neuer Rekord an das Ende von *affectedObject.recordList* angehängt.**[<=]**

G2_N059.650.d.2 - (N059.650)d.2 K_COS
Wenn *affectedObject* vom Typ zyklische EF ist, dann wird ein neuer Rekord am Anfang von *affectedObject.recordList* eingefügt. Wenn dadurch die Anzahl der Listenelemente größer als *affectedObject.maximumNumberOfRecords* wird, genau dann MUSS das letzte Element in *affectedObject.recordList* gelöscht werden.**[<=]**

G2_N059.650.e - (N059.650)e K_COS
Schritt 5: Wenn *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS diese Checksumme auf einen Wert setzen, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist.**[<=]**

G2_N059.650.f - (N059.650)f K_COS
Schritt 6: Wenn der neu angelegte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS dessen Checksumme auf einen Wert setzen, der konsistent zum Inhalt des neuen Rekords ist.**[<=]**

G2_N059.650.g - (N059.650.)g K_COS
Schritt 7: Als Oktettstring des neuen Rekords MUSS *recordData* verwendet werden.**[<=]**

G2_N059.650.h - (N059.650)h K_COS
Schritt 8: Wenn *affectedObject.flagRecordLifeCycleStatus* den Wert True besitzt, dann MUSS der physikalische Wert des *lifeCycleStatus* des neuen Rekords auf "Operational state (active)" gesetzt werden.**[<=]**

(N059.700) Diese Anforderung ist absichtlich leer.

(N059.800) Diese Anforderung ist absichtlich leer.

A_16264 - (N059.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16265 - (N060.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
1. entweder als Trailer MemoryFailure verwendet werden,
2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N060.100 - (N060.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16266 - (N060.200) K_COS
Für die Priorität der Trailer gilt:
1. Die Priorität der Trailer in CosT_8cd ist herstellerspezifisch.
2. Jeder Trailer in CosT_8cd MUSS eine höhere Priorität als UpdateRetryWarning haben.
3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N060.300) Diese Anforderung ist absichtlich leer.

(N060.400) Diese Anforderung ist absichtlich leer.

14.4.3 DEACTIVATE RECORD

Das Kommando DEACTIVATE RECORD deaktiviert ein oder mehrere Listenelemente aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses DEACTIVATE RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses DEACTIVATE RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welche Listenelemente deaktiviert werden bestimmen Rekordnummer und Modus, welche als Parameter in der Kommandonachricht enthalten sind.

14.4.3.1 Use Case Deaktivieren eines Rekords ohne shortFileIdentifier

In dieser Variante enthält die APDU des DEACTIVATE RECORD-Kommandos zwei Parameter:

A_16267 - (N060.500) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16268 - (N060.600) K_externeWelt {K_Karte}
Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.**[<=]**

A_16269 - (N060.700) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_7b1 verwendet werden.

Tabelle 179: CosT_7b1: DEACTIVATE RECORD, ein Rekord, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'06'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'04'	*mode*, Codierung '04' bedeutet "nutze Listenelement P1"

[<=]

14.4.3.2 Use Case Deaktivieren eines Rekords mit shortFileIdentifier

In dieser Variante enthält die APDU des DEACTIVATE RECORD-Kommandos drei Parameter:

A_16270 - (N060.800) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16271 - (N060.900) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16272 - (N061.000) K_externeWelt {K_Karte}
Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.**[<=]**

A_16273 - (N061.100) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_869 verwendet werden.

Tabelle 180: CosT_869: DEACTIVATE RECORD, ein Rekord, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'06'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '04' Codierung '04' bedeutet "nutze Listenelement P1"

[<=]

14.4.3.3 Use Case Deaktivieren aller Rekords ab P1 ohne shortFileIdentifier

In dieser Variante enthält die APDU des DEACTIVATE RECORD-Kommandos zwei Parameter:

A_16274 - (N061.200) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16275 - (N061.300) K_externeWelt {K_Karte}
Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.**[<=]**

A_16276 - (N061.400) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_f8e verwendet werden.

Tabelle 181: CosT_f8e: DEACTIVATE RECORD, alle Rekords ab P1, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'06'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'05'	*mode*, Codierung '05' bedeutet "nutze Listenelemente ab P1"

[<=]

14.4.3.4 Use Case Deaktivieren aller Rekords ab P1 mit shortFileIdentifier

In dieser Variante enthält die APDU des DEACTIVATE RECORD-Kommandos drei Parameter:

A_16277 - (N061.500) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16278 - (N061.600) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16279 - (N061.700) K_externeWelt {K_Karte}
Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.**[<=]**

A_16280 - (N061.800) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_9ec verwendet werden.

Tabelle 182: CosT_9ec: DEACTIVATE RECORD, alle Rekords ab P1, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'06'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '05' Codierung '05' bedeutet "nutze Listenelemente ab P1"

[<=]

14.4.3.5 Antwort der Karte auf Deaktivieren von Rekords

Tabelle 183: CosT_453: DEACTIVATE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Deaktivieren

Tabelle 184: CosT_563: DEACTIVATE RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoRecordLifeCycleStatus	Rekords in ausgewähltem EF besitzen keinen LCS
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement *recordNumber* existiert nicht

Hinweis CosH_648: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16281 - (N061.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das DEACTIVATE RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.3.6 Kommandoabarbeitung innerhalb der Karte

G2_N062.000.a - (N062.000)a K_COS
Das COS MUSS die DEACTIVATE RECORD-Varianten aus
- 14.4.3.1- Use Case Deaktivieren eines Rekords ohne shortFileIdentifier,
- 14.4.3.2- Use Case Deaktivieren eines Rekords mit shortFileIdentifier,
- 14.4.3.3- Use Case Deaktivieren aller Rekords ab P1 ohne shortFileIdentifier und
- 14.4.3.4- Use Case Deaktivieren aller Rekords ab P1 mit shortFileIdentifier

unterstützen.**[<=]**

A_16282 - (N062.000)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere DEACTIVATE RECORD-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16283 - (N062.100)a K_COS
Wenn die APDU des DEACTIVATE RECORD-Kommandos einen *shortFileIdentifier* enthält, genau dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N062.100.a.1 - (N062.100)a.1 K_COS

Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N062.100.a.2 - (N062.100)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N062.100.b.1 - (N062.100)b.1 K_COS
Wenn die APDU des DEACTIVATE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N062.100.b.2 - (N062.100)b.2 K_COS
Wenn die APDU des DEACTIVATE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N062.200 - (N062.200) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N062.300 - (N062.300) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N062.400 - (N062.400) K_COS
Wenn *affectedObject.flagRecordLifeCycleStatus* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer NoRecordLifeCycleStatus terminieren.**[<=]**

G2_N062.500 - (N062.500) K_COS
Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.**[<=]**

G2_N062.600.a - (N062.600)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, dann MUSS der *lifecycleStatus* mit Transaktionsschutz geändert werden.**[<=]**

A_16284 - (N062.600)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.body* mit oder ohne Transaktionsschutz geändert wird.**[<=]**

G2_N062.700 - (N062.700) K_COS
Wenn *mode* = '04' ist und der physikalische Wert von *lifeCycleStatus* des durch *recordNumber* adressierten *record* in *affectedObject.recordList* bereits den Wert "Operational state (deactivated)" besitzt, dann MUSS als Trailer NoError verwendet werden.**[<=]**

A_16285 - (N062.800) K_COS
Die physikalischen Werte von *lifeCycleStatus* der durch *recordNumber* und *mode* adressierten *record* in *affectedObject.recordList* MÜSSEN auf den Wert "Operational state (deactivated)" gesetzt werden.**[<=]**

G2_N062.800.a - (N062.800)a K_COS

Wenn *mode* den Wert '04' besitzt, dann DARF NUR nur das durch*recordNumber* adressierte Listenelement betroffen sein.**[<=]**

G2_N062.800.b - (N062.800)b K_COS
Wenn *mode* den Wert '05' besitzt, dann MUSS das durch*recordNumber* adressierte Listenelement und alle folgenden betroffen sein.**[<=]**

A_16286 - (N062.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16287 - (N063.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
1. entweder als Trailer MemoryFailure verwendet werden,
2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N063.100 - (N063.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16288 - (N063.200) K_COS
Für die Priorität der Trailer gilt:
1. Die Priorität der Trailer in CosT_563 ist herstellerspezifisch.
2. Jeder Trailer in CosT_563 MUSS eine höhere Priorität als UpdateRetryWarning haben.
3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N063.300) Diese Anforderung ist absichtlich leer.

(N063.400) Diese Anforderung ist absichtlich leer.

14.4.4 DELETE RECORD

Das Kommando DELETE RECORD entfernt ein bereits vorhandenes Listenelemente aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses DELETE RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses DELETE RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches Listenelement betroffen ist, bestimmt die Rekordnummer, welche als Parameter in der Kommandonachricht enthalten ist.

Hinweis CosH_5b3: Durch das Entfernen eines Elementes aus einer Liste ändert sich die Adressierung nachfolgender Listenelemente. Enthält beispielsweise eine strukturierte Datei eine Liste mit drei Rekords gemäß '01'→'02'→'03' wird ein READ RECORD-Kommando für Rekord zwei (Kommando APDU = '00B2020400') Listenelement 2 liefern (Response APDU = '029000'). Nach einem DELETE RECORD (Kommando APDU = '800C0204') enthält die Datei folgende Liste: '01'→'03'. EinREAD RECORD-Kommando für Rekord zwei liefert dann: '039000').

14.4.4.1 Use Case Entfernen eines Rekords ohne shortFileIdentifier

In dieser Variante wird ein Rekord aus der Liste *recordList* gelöscht und die betroffene Datei ist zuvor auszuwählen.

A_16293 - (N063.420) K_externeWelt {K_Karte}
Die APDU des DELETE RECORD-Kommandos MUSS einen Parameter enthalten:

a. Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.

[<=]

A_16294 - (N063.422) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_be4 verwendet werden.

Tabelle 185: CosT_be4: DELETE RECORD, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'0C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'04'	Codierung für "nutze Listenelement P1"

[<=]

14.4.4.2 Use Case Entfernen eines Rekords mit shortFileIdentifier

In dieser Variante wird ein Rekord aus der Liste *recordList* gelöscht und die betroffene Datei wird im Kommando adressiert.

A_16295 - (N063.424) K_externeWelt {K_Karte}
Die APDU des DELETE RECORD-Kommandos MUSS zwei Parameter enthalten:

b. Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.

c. Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.

[<=]

A_16296 - (N063.426) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_157 verwendet werden.

Tabelle 186: CosT_157: DELETE RECORD, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'0C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '04' Codierung '04' bedeutet "nutze Listenelement P1"

[<=]

14.4.4.3 Antwort der Karte auf Entfernen eines Rekords

Tabelle 187: CosT_fcd: DELETE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Entfernen eines Rekords

Tabelle 188: CosT_91a: DELETE RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'62 87'	RecordDeactivated	Adressierter Rekord ist deaktiviert
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement *recordNumber* existiert nicht

Hinweis CosH_684: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16297 - (N063.428) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das DELETE RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.4.4 Kommandoabarbeitung innerhalb der Karte

G2_N063.430.a - (N063.430)a K_COS
Das COS MUSS die DELETE RECORD-Varianten aus
- 14.4.4.1- Use Case Entfernen eines Rekords ohne shortFileIdentifier und
- 14.4.4.2- Use Case Entfernen eines Rekords mit shortFileIdentifier

unterstützen.**[<=]**

A_16298 - (N063.430)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere DELETE RECORD-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16299 - (N063.432)a K_COS
Wenn die APDU des DELETE RECORD-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N063.432.a.1 - (N063.432)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N063.432.a.2 - (N063.432)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N063.432.b.1 - (N063.432)b.1 K_COS
Wenn die APDU des DELETE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N063.432.b.2 - (N063.432)b.2 K_COS
Wenn die APDU des DELETE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N063.434 - (N063.434) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N063.436 - (N063.436) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N063.438 - (N063.438) K_COS
Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.**[<=]**

G2_N063.440 - (N063.440) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand "Operational state (deactivated)" hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.**[<=]**

A_16300 - (N063.442) K_COS
Wenn *affectedObject.recordList* oder der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und den Daten, die sie schützt,
1. DARF NICHT zum Abbruch des Kommandos führen.
2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

A_18351 - (N063.444)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, dann MUSS *affectedObject.recordList* und deren eventuell vorhandene Checksumme mit Transaktionsschutz geändert werden.**[<=]**

A_18352 - (N063.444)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass *affectedObject.recordList* und deren eventuell vorhandene Checksumme mit oder ohne Transaktionsschutz geändert wird.**[<=]**

G2_N063.446.a - (N063.446)a K_COS
Der durch *recordNumber* adressierte *record* MUSS aus *affectedObject. recordList* entfernt werden.**[<=]**

A_16301 - (N063.446)b K_COS

Wenn *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist.**[<=]**

A_16302 - (N063.448) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16303 - (N063.450) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
1. entweder als Trailer MemoryFailure verwendet werden,
2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N063.452 - (N063.452) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16304 - (N063.454) K_COS
Für die Priorität der Trailer gilt:
1. Die Priorität der Trailer in CosT_91a ist herstellerspezifisch.
2. Jeder Trailer in CosT_91a MUSS eine höhere Priorität als UpdateRetryWarning haben.
3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.4.5 ERASE RECORD

Das Kommando ERASE RECORD setzt jedes Oktett eines bereits vorhandenen Listenelementes in *recordList* eines strukturierten EF auf den Wert '00'. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden diesesERASE RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses ERASE RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches Listenelement betroffen ist, bestimmt die Rekordnummer, welche als Parameter in der Kommandonachricht enthalten ist.

14.4.5.1 Use Case Löschen eines Rekordinhaltes ohne shortFileIdentifier

In dieser Variante werden Daten in einem Rekord durch Oktette mit dem Wert '00' ersetzt und die betroffene Datei ist zuvor auszuwählen. Der Rekord selbst verbleibt in der Liste *recordList*. Die APDU des ERASE RECORD-Kommandos enthält einen Parameter:

A_16305 - (N063.500) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16306 - (N063.600) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_e66 verwendet werden.

Tabelle 189: CosT_e66: ERASE RECORD, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4], adressierter Rekord bleibt erhalten
INS	'0C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'04'	Codierung für "nutze Listenelement P1"

[<=]

14.4.5.2 Use Case Löschen eines Rekordinhaltes mit shortFileIdentifier

In dieser Variante werden Daten in einem Rekord durch Oktette mit dem Wert '00' ersetzt und die betroffene Datei wird im Kommando adressiert. Der Rekord selbst verbleibt in der Liste *recordList*. Die APDU des ERASE RECORD-Kommandos enthält zwei Parameter:

A_16307 - (N063.700) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16308 - (N063.800) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16309 - (N063.900) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_124 verwendet werden.

Tabelle 190: CosT_124: ERASE RECORD, mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4], adressierter Rekord bleibt erhalten
INS	'0C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '04' Codierung '04' bedeutet "nutze Listenelement P1"

[<=]

14.4.5.3 Antwort der Karte auf Löschen eines Rekordinhaltes

Tabelle 191: CosT_e14: ERASE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Löschen eines Rekordinhaltes

Tabelle 192: CosT_173: ERASE RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'62 87'	RecordDeactivated	Adressierter Rekord ist deaktiviert
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement *recordNumber* existiert nicht

Hinweis CosH_d9b: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16310 - (N064.000) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das ERASE RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.5.4 Kommandoabarbeitung innerhalb der Karte

G2_N064.100.a - (N064.100)a K_COS
Das COS MUSS die ERASE RECORD-Varianten aus
- 14.4.5.1- Use Case Löschen eines Rekordinhaltes ohne shortFileIdentifier und
- 14.4.5.2- Use Case Löschen eines Rekordinhaltes mit shortFileIdentifier

unterstützen.**[<=]**

A_16311 - (N064.100)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere ERASE RECORD-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16312 - (N064.200)a K_COS
Wenn die APDU des ERASE RECORD-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N064.200.a.1 - (N064.200)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N064.200.a.2 - (N064.200)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N064.200.b.1 - (N064.200)b.1 K_COS
Wenn die APDU des ERASE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N064.200.b.2 - (N064.200)b.2 K_COS
Wenn die APDU des ERASE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N064.300 - (N064.300) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N064.400 - (N064.400) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N064.500 - (N064.500) K_COS
Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.**[<=]**

G2_N064.600 - (N064.600) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand "Operational state (deactivated)" hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.**[<=]**

A_16313 - (N064.700) K_COS
Das COS MUSS die Art des Transaktionsschutzes festlegen.**[<=]**

G2_N064.700.a - (N064.700)a K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, dann MUSS der Inhalt des adressierten Rekords mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.recordList* oder der Checksumme des durch *recordNumber* adressierten *record* in *affectedObject.recordList* umfassen, sofern diese vorhanden sind.**[<=]**

A_16314 - (N064.700)b K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass der Inhalt des adressierten Rekords und dessen eventuell vorhandene Checksumme mit oder ohne Transaktionsschutz geändert wird.**[<=]**

A_16315 - (N064.710) K_COS
Falls *affectedObject.recordList* oder der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und den Daten, die sie schützt,
a. DARF NICHT zum Kommandoabbruch führen.
b. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

A_16316 - (N064.800) K_COS
Das COS MUSS den Rekordinhalt durch folgende Aktionen löschen:**[<=]**

G2_N064.800.a - (N064.800)a K_COS
Das COS MUSS alle Oktette im Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* auf den Wert '00' setzen.**[<=]**

A_16317 - (N064.800)b K_COS
Wenn *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist.**[<=]**

A_16318 - (N064.800)c K_COS
Wenn der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS dessen Checksumme auf einen Wert gesetzt werden, der konsistent zum neuen Inhalt von diesem Rekord ist.**[<=]**

A_16319 - (N064.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16320 - (N065.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
1. entweder als Trailer MemoryFailure verwendet werden,
2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N065.100 - (N065.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16321 - (N065.200) K_COS
Für die Priorität der Trailer gilt:
1. Die Priorität der Trailer in CosT_173 ist herstellerspezifisch.
2. Jeder Trailer in CosT_173 MUSS eine höhere Priorität als UpdateRetryWarning haben.
3. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N065.300) Diese Anforderung ist absichtlich leer.

(N065.400) Diese Anforderung ist absichtlich leer.

14.4.6 READ RECORD

Das Kommando READ RECORD dient dem Auslesen (des Anfangs) eines Listenelementes aus *recordList* eines strukturierten EF. Das betroffene strukturierte EF wird vor der Leseoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses READ RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses READ RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches Listenelement (oder dessen Anfang) ausgelesen wird, bestimmen Rekordnummer und Länge, die als Parameter in der Kommandonachricht enthalten sind.

14.4.6.1 Use Case Lesen ohne shortFileIdentifier in strukturierten EF

In dieser Variante enthält die APDU des READ RECORD-Kommandos zwei Parameter:

A_16324 - (N065.500) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16325 - (N065.600) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16326 - (N065.700) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_dd7 verwendet werden.

Tabelle 193: CosT_dd7: READ RECORD ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'B2'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'04'	Codierung für "nutze Listenelement P1"
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.4.6.2 Use Case Lesen mit shortFileIdentifier in strukturierten EF

In dieser Variante enthält die APDU des READ RECORD-Kommandos drei Parameter:

A_16327 - (N065.800) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16328 - (N065.900) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16329 - (N066.000) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16330 - (N066.100) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_b99 verwendet werden.

Tabelle 194: CosT_b99: READ RECORD mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'B2'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '04' Codierung '04' bedeutet "nutze Listenelement P1"
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.4.6.3 Antwort der Karte auf Lesen in strukturierten EF

Tabelle 195: CosT_bb0: READ RECORD Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'		Ausgelesene Daten
Trailer	**Inhalt**	**Beschreibung**
'62 81'	CorruptDataWarning	Möglicherweise sind die Antwortdaten korrupt
'62 82'	EndOfRecordWarning	Mittels Ne mehr Daten angefordert, als vorhanden sind
'90 00'	NoError	Erfolgreiche Leseoperation

Tabelle 196: CosT_b05: READ RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'62 87'	RecordDeactivated	Adressierter Rekord ist deaktiviert
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement *recordNumber* existiert nicht

Hinweis CosH_758: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16331 - (N066.200) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das READ RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.6.4 Kommandoabarbeitung innerhalb der Karte

G2_N066.300.a - (N066.300)a K_COS
Das COS MUSS die READ RECORD-Varianten aus
- 14.4.6.1- Use Case Lesen ohne shortFileIdentifier in strukturierten EF und
- 14.4.6.2- Use Case Lesen mit shortFileIdentifier in strukturierten EF

unterstützen.**[<=]**

A_16332 - (N066.300)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere READ RECORD-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16333 - (N066.400)a K_COS
Wenn die APDU des READ RECORD-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N066.400.a.1 - (N066.400)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N066.400.a.2 - (N066.400)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N066.400.b.1 - (N066.400)b.1 K_COS
Wenn die APDU des READ RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N066.400.b.2 - (N066.400)b.2 K_COS
Wenn die APDU des READ RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N066.500 - (N066.500) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N066.600 - (N066.600) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N066.700 - (N066.700) K_COS
Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.**[<=]**

G2_N066.800 - (N066.800) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand "Operational state (deactivated)" hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.**[<=]**

A_16334 - (N066.900) K_COS
Wenn *affectedObject.flagChecksum* den Wert True hat und die Daten des adressierten Listenelementes inkonsistent zur Checksumme sind, genau dann MUSS
a. entweder als Trailer CorruptDataWarning gewählt werden,
b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N067.000 - (N067.000) K_COS
Wenn das LeFeld der Kommando-APDU keine WildCard enthält und *length* größer als die Länge des adressierten Listenelementes in *affectedObject.numberOfOctet* ist, genau dann MUSS als Trailer EndOfRecordWarning gewählt werden.**[<=]**

G2_N067.100 - (N067.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

G2_N067.200.a - (N067.200)a K_COS
Aus dem Oktettstring des adressierten *record* MÜSSEN das erste Oktett und die nachfolgenden Oktette in das Datenfeld der Antwortnachricht übernommen werden.**[<=]**

G2_N067.200.b - (N067.200)b K_COS
Es DÜRFEN NICHT mehr Oktette übernommen werden, als durch *length* angegeben.**[<=]**

G2_N067.200.c - (N067.200)c K_COS
Die Übernahme MUSS am Ende von *record* stoppen.**[<=]**

A_16335 - (N067.300)a K_TST

Die Priorität der Trailer in CosT_b05 MUSS herstellerspezifisch sein.**[<=]**

G2_N067.300.b - (N067.300)b K_COS
Jeder Trailer in CosT_b05 MUSS eine höhere Priorität als CorruptDataWarning haben.**[<=]**

G2_N067.300.c - (N067.300)c K_COS
CorruptDataWarning MUSS eine höhere Priorität als EndOfRecordWarning haben.**[<=]**

G2_N067.300.d - (N067.300)d K_COS
EndOfRecordWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N067.400) Diese Anforderung ist absichtlich leer.

(N067.500) Diese Anforderung ist absichtlich leer.

14.4.7 SEARCH RECORD

Das Kommando SEARCH RECORD sucht in den Listenelementen von *recordList* eines strukturierten EF nach einem Muster, welches im Datenfeld der Kommandonachricht übergeben wird. Die Antwortdaten enthalten die Nummern der Rekords, welche das Muster enthalten. Das betroffene, strukturierte EF wird vor der Suchoperation ausgewählt. Dies geschieht entweder vor dem Senden dieses SEARCH RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses SEARCH RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Bei welchem Listenelement in *recordList* die Suche startet, wird durch die Rekordnummer bestimmt, die als Parameter in der Kommandonachricht enthalten ist.

14.4.7.1 Use Case Suchen ohne shortFileIdentifier in strukturierten EF

In dieser Variante enthält die APDU des SEARCH RECORD-Kommandos drei Parameter:

A_16337 - (N067.600) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das Listenelement, welches als erstes von der Suche betroffen ist. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16338 - (N067.700) K_externeWelt {K_Karte}
Der Parameter *searchString* enthält das Muster, nach welchem in den Oktettstrings der Listenelemente gesucht wird. Der Parameter *searchString* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *searchString* MUSS im Intervall [1, 255] Oktett sein.**[<=]**

A_16339 - (N067.800) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16340 - (N067.900) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_597 verwendet werden.

Tabelle 197: CosT_597: SEARCH RECORD ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A2'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'04'	Codierung für "suche in Listenelement P1 und allen folgenden"
Data	'XX…XX'	*searchString*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.4.7.2 Use Case Suchen mit shortFileIdentifier in strukturierten EF

In dieser Variante enthält die APDU des SEARCH RECORD-Kommandos vier Parameter:

A_16341 - (N068.000) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16342 - (N068.100) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das Listenelement, welches als erstes von der Suche betroffen ist. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16343 - (N068.200) K_externeWelt {K_Karte}
Der Parameter *searchString* enthält das Muster, nach welchem in den Oktettstrings der Listenelemente gesucht wird. Der Parameter *searchString* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *searchString* MUSS im Intervall [1, 255] Oktett sein.**[<=]**

A_16344 - (N068.300) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus dem in (N027.000) definierten Bereich gewählt werden.**[<=]**

A_16345 - (N068.400) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_771 verwendet werden.

Tabelle 198: CosT_771: SEARCH RECORD mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'A2'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '04' '04' bedeutet: "suche in Listenelement P1 und allen folgenden"
Data	'XX…XX'	*searchString*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.4.7.3 Antwort der Karte auf Suchen in strukturierten EF

Tabelle 199: CosT_13d: SEARCH RECORD Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*rspData*	Nummern der Listenelemente, in denen das Muster gefunden wurde
Trailer	**Inhalt**	**Beschreibung**
'62 81'	CorruptDataWarning	Möglicherweise sind Antwortdaten korrupt
'62 82'	UnsuccessfulSearch	Erfolglose Suche in adressierten Rekords
'90 00'	NoError	Erfolgreiche Suchoperation

Tabelle 200: CosT_8b8: SEARCH RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* ausgewähltes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement *recordNumber* existiert nicht

Hinweis CosH_9cc: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16346 - (N068.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das SEARCH RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.7.4 Kommandoabarbeitung innerhalb der Karte

G2_N068.600.a - (N068.600)a K_COS

Das COS MUSS die SEARCH RECORD-Varianten aus
- 14.4.7.1- Use Case Suchen ohne shortFileIdentifier in strukturierten EF und
- 14.4.7.2- Use Case Suchen mit shortFileIdentifier in strukturierten EF

unterstützen.**[<=]**

A_16347 - (N068.600)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere SEARCH RECORD-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16348 - (N068.700)a K_COS
Wenn die APDU des SEARCH RECORD-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N068.700.a.1 - (N068.700)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N068.700.a.2 - (N068.700)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N068.700.b.1 - (N068.700)b.1 K_COS
Wenn die APDU des SEARCH RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N068.700.b.2 - (N068.700)b.2 K_COS
Wenn die APDU des SEARCH RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N068.800 - (N068.800) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N068.900 - (N068.900) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N069.000 - (N069.000) K_COS
Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.**[<=]**

G2_N069.100 - (N069.100) K_COS
Im Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* und allen folgenden Elementen der Liste MUSS nach dem Muster *searchString* gesucht werden.**[<=]**

G2_N069.200 - (N069.200) K_COS
Die Suche in einem Listenelement MUSS genau dann erfolgreich sein, wenn
a. der physikalische Wert des *lifeCycleStatus* des Listenelementes den Wert "Operational

state (active)" hat UND
b. *searchString* im Oktettstring des Listenelementes vollständig enthalten ist.**[<=]**

G2_N069.300 - (N069.300) K_COS
Wenn die Suche in einem Listenelement erfolgreich war, dann MUSS die Nummer des Rekords (siehe (N007.600)a) in einem Oktett (gemäß I2OS(*recordNumber*, 1)) codiert zum Datenfeld *rspData* der Antwortnachricht hinzugefügt werden.**[<=]**

G2_N069.400 - (N069.400) K_COS
Die Oktette im Datenfeld *rspData* MÜSSEN aufsteigend sortiert sein.**[<=]**

A_16349 - (N069.500) K_COS
Wenn *affectedObject.flagChecksum* den Wert True hat und die Daten wenigstens eines adressierten Listenelementes inkonsistent zur Checksumme sind, genau dann MUSS
a. entweder als Trailer CorruptDataWarning gewählt werden,
b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N069.600 - (N069.600) K_COS
Wenn das Datenfeld *rspData* leer ist, das heißt die Suche war in keinem der adressierten Listenelemente erfolgreich, genau dann MUSS als Trailer UnsuccessfulSearch verwendet werden.**[<=]**

G2_N069.700 - (N069.700) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

G2_N069.710 - (N069.710) K_COS
Für das Datenfeld *rspData* der Antwortnachricht gilt: Wenn OctetLength(*rspData*) kleiner Nr gemäß (N027.200) ist,
a. dann *rspData* = *rspData*,
b. sonst *rspData* = Extract_MSByte(*rspData*, Nr).**[<=]**

A_16350 - (N069.800)a K_TST
Die Priorität der Trailer in CosT_8b8 MUSS herstellerspezifisch sein.**[<=]**

G2_N069.800.b - (N069.800)b K_COS
Jeder Trailer in CosT_8b8 MUSS eine höhere Priorität als CorruptDataWarning haben.**[<=]**

G2_N069.800.c - (N069.800)c K_COS
CorruptDataWarning MUSS eine höhere Priorität als UnsuccessfulSearch haben.**[<=]**

G2_N069.800.d - (N069.800)d K_COS
UnsuccessfulSearch MUSS eine höhere Priorität als NoError haben.**[<=]**

(N069.900) Diese Anforderung ist absichtlich leer.

(N070.000) Diese Anforderung ist absichtlich leer.

14.4.8 UPDATE RECORD

Das Kommando UPDATE RECORD ersetzt den Oktettstring eines bereits vorhandenen Listenelementes in *recordList* eines strukturierten EF durch Daten, die im Datenfeld der Kommandonachricht enthalten sind. Das betroffene strukturierte EF wird vor der Schreiboperation ausgewählt. Dies geschieht entweder vor dem Senden dieses UPDATE RECORD-Kommandos durch eine Select-Operation (SELECT-Kommando oder Kommando mit *shortFileIdentifier*), oder innerhalb dieses UPDATE RECORD-Kommandos, falls diesem ein *shortFileIdentifier* als Parameter mitgeliefert wurde. Welches

Listenelement in *recordList* ersetzt wird, bestimmt die Rekordnummer, die als Parameter in der Kommandonachricht enthalten ist.

14.4.8.1 Use Case Rekordinhalt schreiben, ohne shortFileIdentifier

Die APDU des UPDATE RECORD-Kommandos enthält zwei Parameter:

A_16353 - (N070.100) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16354 - (N070.200) K_externeWelt {K_Karte}
Der Parameter *newData* enthält die neuen Daten, welche den Oktettstring des adressierten *record* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N007.700)a definierten Bereich gewählt werden.**[<=]**

A_16355 - (N070.300) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_dc9 verwendet werden.

Tabelle 201: CosT_dc9: UPDATE RECORD, ohne shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'DC'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'04'	Codierung für "nutze Listenelement P1"
Data	'XX…XX'	*newData*

[<=]

14.4.8.2 Use Case Rekordinhalt schreiben, mit shortFileIdentifier

Die APDU des UPDATE RECORD-Kommandos enthält drei Parameter:

A_16356 - (N070.400) K_externeWelt {K_Karte}
Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.**[<=]**

A_16357 - (N070.500) K_externeWelt {K_Karte}
Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600)a gewählt werden.**[<=]**

A_16358 - (N070.600) K_externeWelt {K_Karte}
Der Parameter *newData* enthält die neuen Daten, welche den Oktettstring des adressierten *record* ersetzen. Der Parameter *newData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *newData* MUSS aus dem in (N007.700)a definierten Bereich gewählt werden.**[<=]**

A_16359 - (N070.700) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_df8 verwendet werden.

Tabelle 202: CosT_df8: UPDATE RECORD mit shortFileIdentifier

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'DC'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	*recordNumber*
P2	'XX'	8 *shortFileIdentifier* + *mode*, das heißt (*shortFileIdentifier* << 3) + '04' Codierung '04' bedeutet "nutze Listenelement P1"
Data	'XX…XX'	*newData*

[<=]

14.4.8.3 Antwort der Karte auf Schreiben in strukturierten EF

Tabelle 203: CosT_d43: UPDATE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Schreibvorgang

Tabelle 204: CosT_aff: UPDATE RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'62 87'	RecordDeactivated	Adressierter Rekord ist deaktiviert
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'67 00'	WrongRecordLength	*newData* hat nicht die richtige Länge
'69 81'	WrongFileType	Ausgewähltes EF ist nicht strukturiert
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 86'	NoCurrentEF	Es ist kein EF ausgewählt
'6A 82'	FileNotFound	Per *shortFileIdentifier* ausgewähltes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement *recordNumber* existiert nicht
'6A 84'	OutOfMemory	Zu viele Oktette in *newData*

Hinweis CosH_789: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16360 - (N070.800) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das UPDATE RECORD-Kommando zusätzliche Trailer verwendet.**[<=]**

14.4.8.4 Kommandoabarbeitung innerhalb der Karte

G2_N070.900.a - (N070.900)a K_COS
Das COS MUSS die UPDATE RECORD-Varianten aus
- 14.4.8.1- Use Case Rekordinhalt schreiben, ohne shortFileIdentifier und
- 14.4.8.2- Use Case Rekordinhalt schreiben, mit shortFileIdentifier

unterstützen.**[<=]**

A_16361 - (N070.900)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere UPDATE RECORD-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16362 - (N071.000)a K_COS
Wenn die APDU des UPDATE RECORD-Kommandos einen *shortFileIdentifier* enthält, dann MUSS innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht werden.**[<=]**

G2_N071.000.a.1 - (N071.000)a.1 K_COS
Wenn die Suche erfolgreich verlief, dann MUSS
i. *affectedObject* auf dieses EF gesetzt werden und
ii. *currentEF* auf dieses EF gesetzt werden.**[<=]**

G2_N071.000.a.2 - (N071.000)a.2 K_COS
Wenn die Suche nicht erfolgreich verlief, genau dann MUSS
i. *currentEF* unverändert bleiben und
ii. das Kommando mit dem Trailer FileNotFound terminieren.**[<=]**

G2_N071.000.b.1 - (N071.000)b.1 K_COS
Wenn die APDU des UPDATE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren.**[<=]**

G2_N071.000.b.2 - (N071.000)b.2 K_COS
Wenn die APDU des UPDATE RECORD-Kommandos keinen *shortFileIdentifier* enthält und *currentEF* (siehe (N029.900)m) auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.**[<=]**

G2_N071.100 - (N071.100) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N071.200 - (N071.200) K_COS
Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.**[<=]**

G2_N071.300 - (N071.300) K_COS
Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedObject.recordList* ist, genau dann MUSS das Kommando mit dem Trailer RecordNotFound terminieren.**[<=]**

G2_N071.400 - (N071.400) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zustand "Operational state (deactivated)" hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.**[<=]**

(N071.500) Diese Anforderung ist absichtlich leer.

(N071.600) Diese Anforderung ist absichtlich leer.

(N071.700) Diese Anforderung ist absichtlich leer.

A_16363 - (N071.750) K_COS
Das COS MUSS folgende Schritte ausführen:**[<=]**

A_16364 - (N071.750)a K_COS
Schritt 1: Das COS MUSS die Anzahl der Oktette in *recordData* mit *affectedObject.maximumRecordLength* vergleichen.**[<=]**

G2_N071.750.a.1 - (N071.750)a.1
Wenn *affectedObject* vom Typ linear fixes EF ist und die Anzahl Oktette in *newData* ungleich *affectedObject.maximumRecordLength* ist, dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.**[<=]**

G2_N071.750.a.2 - (N071.750)a.2 K_COS
Wenn *affectedObject* vom Typ zyklisches EF ist und die Anzahl Oktette in *newData* ungleich *affectedObject.maximumRecordLength* ist, dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.**[<=]**

G2_N071.750.a.3.i - (N071.750)a.3.i K_COS
Wenn *affectedObject* vom Typ linear variables EF ist und die Anzahl Oktette in *newData* größer als *affectedObject.maximumRecordLength* ist, dann MUSS das Kommando mit dem Trailer WrongRecordLength terminieren.**[<=]**

G2_N071.750.a.3.ii - (N071.750)a.3.ii K_COS
Wenn *affectedObject* vom Typ linear variables EF ist die Anzahl Oktette in den Oktettstrings aller *record* von *recordList* nach durchgeführter Ersetzung größer als *affectedObject.numberOfOctet* wäre, genau dann MUSS das Kommando mit dem Trailer OutOfMemory terminieren.**[<=]**

A_16365 - (N071.750)b K_COS
Schritt 2: Das COS MUSS die Art des Transaktionsschutzes festlegen.**[<=]**

G2_N071.750.b.1 - (N071.750)b.1 K_COS
Wenn *affectedObject.flagTransactionMode* den Wert True hat, dann MUSS der Rekordinhalt mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.recordList* oder der Checksumme des geänderten Rekords umfassen, sofern diese vorhanden sind.**[<=]**

A_16366 - (N071.750)b.2 K_TST
Wenn *affectedObject.flagTransactionMode* den Wert False hat, dann MUSS es für die funktionale Eignung zulässig sein, dass der Inhalt des adressierten Rekords und dessen eventuell vorhandene Checksumme mit oder ohne Transaktionsschutz geändert wird.**[<=]**

A_16367 - (N071.750)c K_COS
Schritt 3: Wenn *affectedObject.recordList* oder der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS das COS auf eine der in diesem Punkt genannten Arten reagieren: Eine Inkonsistenz zwischen dieser Checksumme und *affectedObject.recordList*
1. DARF NICHT zum Abbruch des Kommandos führen.
2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

A_16368 - (N071.750)d K_COS
Schritt 4: Das COS MUSS den Inhalt des betroffenen Rekords wie folgt ändern:**[<=]**

G2_N071.750.d.1 - (N071.750)d.1 K_COS

Der Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* wird durch *newData* ersetzt.**[<=]**

G2_N071.750.d.2 - (N071.750)d.2 K_COS
Falls *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS diese Checksumme auf einen Wert gesetzt werden, der konsistent zum geänderten Inhalt von *affectedObject.recordList* ist.**[<=]**

G2_N071.750.d.3 - (N071.750)d.3 K_COS
Falls der durch *recordNumber* adressierte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS dessen Checksumme auf einen Wert gesetzt werden, der konsistent zum neuen Inhalt von diesem Rekord ist.**[<=]**

A_16369 - (N071.800) K_COS
Falls *affectedObject* vom Typ linear variables EF ist, dann MÜSSEN alle folgenden Fälle unterstützt werden: *newData* enthält im Vergleich zum alten Inhalt von *record*, der durch *recordNumber* adressiert wird**[<=]**

G2_N071.800.a - (N071.800)a K_COS
weniger Oktette,**[<=]**

G2_N071.800.b - (N071.800)b K_COS
gleich viele Oktette, oder**[<=]**

G2_N071.800.c - (N071.800)c K_COS
mehr Oktette.**[<=]**

A_16370 - (N071.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16371 - (N072.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
1. entweder als Trailer MemoryFailure verwendet werden,
2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N072.100 - (N072.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16372 - (N072.200) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_aff ist herstellerspezifisch.
b. Jeder Trailer in CosT_aff MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

(N072.300) Diese Anforderung ist absichtlich leer.

(N072.400) Diese Anforderung ist absichtlich leer

14.4.9 WRITE RECORD

A_16373 - (N072.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das WRITE RECORD-Kommando gemäß [ISO/IEC 7816-4]
a. unterstützt oder
b. ablehnt.**[<=]**

14.5 Zugriff auf Datenobjekte

Hinweis CosH_dfd. Die in diesem Kapitel behandelten Kommandos sind nicht verpflichtend, siehe auch 8.7.

14.5.1 GET DATA

A_16374 - (N072.600) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das GET DATA-Kommando gemäß [ISO/IEC 7816-4]
a. unterstützt oder
b. ablehnt.**[<=]**

14.5.2 PUT DATA

A_16375 - (N072.700) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das PUT DATA-Kommando gemäß [ISO/IEC 7816-4]
a. unterstützt oder
b. ablehnt.**[<=]**

14.6 Zugriff auf Passwortobjekte

Alle Kommandos dieses Kapitels benutzen bei der Kommandobearbeitung Passwortobjekte gemäß 8.4 oder 8.5. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in den Kommandodaten enthalten ist. Für diese Passwortreferenz gilt:

A_16376 - (N072.800) K_externeWelt {K_Karte}
Der Parameter *passwordReference* besteht aus den zwei Teilen *location* und *identifier*. *location* zeigt an, ob ein globales oder DF-spezifisches Passwort von der Aktion betroffen ist. Als Wert für *location* MUSS ein Element der Menge {0, 128} = {'00', '80'} verwendet werden. Dabei gilt:

d. Der Wert *location* = '00' MUSS verwendet werden, wenn ein globales Passwortobjekt betroffen ist (siehe (N020.800)).
e. Der Wert *location* = '80' MUSS verwendet werden, wenn ein DF-spezifisches Passwortobjekt betroffen ist (siehe (N020.900)).
f. Der Parameter *identifier* bestimmt das betroffene Passwortobjekt. Der Wert von *identifier* MUSS konform zu (N015.000)a gewählt werden.
g. Der Parameter *passwordReference* MUSS in einem Oktett mit folgendem Wert codiert werden: *passwordReference* = *location* + *identifier*.

[<=]

14.6.1 CHANGE REFERENCE DATA

Das Kommando CHANGE REFERENCE DATA ersetzt das Attribut *secret* eines Passwortobjektes durch Daten, die im Datenfeld der Kommandonachricht enthalten sind. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist. Dieses Dokument spezifiziert die folgenden Varianten:

- Das Kommandodatenfeld enthält das alte und neue Benutzergeheimnis.
- Das Kommandodatenfeld enthält nur das neue Benutzergeheimnis.

14.6.1.1 Use Case Ändern eines Benutzergeheimnisses

In dieser Variante enthält die APDU des CHANGE REFERENCE DATA-Kommandos drei Parameter:

A_16377 - (N072.900) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16378 - (N073.000) K_externeWelt {K_Karte}
Der Parameter *oldSecret* enthält das alte Benutzergeheimnis.**[<=]**

A_16379 - (N073.100) K_externeWelt {K_Karte}
Der Parameter *newSecret* enthält das neue Benutzergeheimnis.**[<=]**

A_16380 - (N073.200) K_externeWelt {K_Karte}
Die Parameter *oldSecret* und *newSecret* MÜSSEN gemäß (N008.100) codiert sein.**[<=]**

A_16381 - (N073.300) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_204 verwendet werden.

Tabelle 205: CosT_204: CHANGE REFERENCE DATA mit altem und neuem Benutzergeheimnis

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'24'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Data enthält altes und neues Benutzergeheimnis
P2	'XX'	*passwordReference*
Data	'XX…XX'	*oldSecret* || *newSecret*

[<=]

14.6.1.2 Use Case Setzen eines Benutzergeheimnisses

In dieser Variante enthält die APDU des CHANGE REFERENCE DATA-Kommandos zwei Parameter:

A_16382 - (N073.400) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16383 - (N073.500) K_externeWelt {K_Karte}
Der Parameter *newSecret* enthält das neue Benutzergeheimnis.**[<=]**

A_16384 - (N073.600) K_externeWelt {K_Karte}
Der Parameter *newSecret* MUSS gemäß (N008.100) codiert sein.**[<=]**

A_16385 - (N073.700) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_95c verwendet werden.

Tabelle 206: CosT_95c: CHANGE REFERENCE DATA, nur neues Benutzergeheimnis

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'24'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'01'	Data enthält neues Benutzergeheimnis
P2	'XX'	*passwordReference*
Data	'XX…XX'	*newSecret*

[<=]

14.6.1.3 Antwort der Karte auf Ändern eines Benutzergeheimnisses

Tabelle 207: CosT_24b: CHANGE REFERENCE DATA Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	*oldSecret* ist falsch
'90 00'	NoError	Erfolgreiches Ändern des Benutzergeheimnisses

Tabelle 208: CosT_1d0: CHANGE REFERENCE DATA Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	PasswordBlocked	Abgelaufener Fehlbedienungszähler
'69 85'	LongPassword	*newData* enthält ein zu langes Passwort
'69 85'	ShortPassword	*newData* enthält ein zu kurzes Passwort
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_371: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16386 - (N073.800) K_TST

Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das CHANGE REFERENCE DATA-Kommando zusätzliche Trailer verwendet.**[<=]**

14.6.1.4 Kommandoabarbeitung innerhalb der Karte

G2_N073.900 - (N073.900) K_COS
Das COS MUSS die CHANGE REFERENCE DATA-Varianten aus
- 14.6.1.1- Use Case Ändern eines Benutzergeheimnisses und
- 14.6.1.2- Use Case Setzen eines Benutzergeheimnisses

unterstützen.**[<=]**

(N074.000) Diese Anforderung ist absichtlich leer.

A_16387 - (N074.100) K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere CHANGE REFERENCE DATA-Varianten
a. unterstützt oder
b. ablehnt.**[<=]**

G2_N074.200 - (N074.200) K_COS
Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

(N074.250) Diese Anforderung ist absichtlich leer.

G2_N074.300 - (N074.300) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N074.400 - (N074.400) K_COS
Wenn *affectedObject.retryCounter* den Wert null hat, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.**[<=]**

G2_N074.500.a - (N074.500)a K_COS
Wenn die in *newSecret* codierte Ziffernfolge für das Attribut *secret* des Passwortobjektes eine Länge hat, die kleiner als *affectedObject.minimumLength* ist, genau dann MUSS das Kommando mit dem Trailer ShortPassword terminieren.**[<=]**

G2_N074.500.b - (N074.500)b K_COS
Wenn die in *newSecret* codierte Ziffernfolge für das Attribut *secret* des Passwortobjektes eine Länge hat, die größer als *affectedObject.maximumLength* ist, genau dann MUSS das Kommando mit dem Trailer LongPassword terminieren.**[<=]**

(N074.600) Diese Anforderung ist absichtlich leer.

Hinweis CosH_b4d: Die Aussage in (N074.700)a.1 bezieht sich bewusst nur auf den momentan aktiven logischen Kanal. Daraus folgt, dass Sicherheitszustände in anderen logischen Kanälen von der dort beschriebenen Aktion unberührt bleiben.

A_16388 - (N074.700) K_COS
Wenn das Datenfeld der Kommandonachricht *oldSecret* enthält, genau dann MUSS das Attribut *affectedObject.secret* mit *oldSecret* verglichen werden.**[<=]**

G2_N074.700.a - (N074.700)a K_COS
Wenn der Vergleich fehlschlägt, genau dann MUSS

1. der Sicherheitszustand im *channelContext* des momentan aktiven logischen Kanals mittels
   clearPasswordStatus(*affectedObject*) zurückgesetzt werden,
2. *affectedObject.retryCounter* um eins dekrementiert werden und
3. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert 'F' gesetzt werden,
   wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*.

[<=]

G2_N074.700.b - (N074.700)b K_COS
Wenn der Vergleich erfolgreich ist, genau dann MUSS das Attribut *affectedObject.retryCounter* auf den Wert *affectedObject.startRetryCounter* gesetzt werden.**[<=]**

G2_N074.700.c - (N074.700)c K_COS
Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.**[<=]**

G2_N074.710 - (N074.710) K_COS
Das Attribut *affectedObject.secret* MUSS auf den in *newSecret* codierten Wert gesetzt werden.**[<=]**

G2_N074.720 - (N074.720) K_COS
Das Attribut *affectedObject.transportStatus* MUSS auf den Wert *regularPassword* geändert werden (siehe (N009.500)a).**[<=]**

G2_N074.800 - (N074.800) K_COS
Alle persistenten Änderungen in (N074.700)b, (N074.710) und (N074.720) MÜSSEN mit Transaktionsschutz ausgeführt werden.**[<=]**

(N074.900) Diese Anforderung ist absichtlich leer.

A_16389 - (N075.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N075.100 - (N075.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16390 - (N075.200)a K_TST
Die Priorität der Trailer in CosT_1d0 MUSS herstellerspezifisch sein.**[<=]**

G2_N075.200.b - (N075.200)b K_COS
Jeder Trailer in CosT_1d0 MUSS eine höhere Priorität als WrongSecretWarning haben.**[<=]**

G2_N075.300.d - (N075.300)d K_COS
WrongSecretWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für konditionale Änderung von *affectedObject.retryCounter* und die Änderung von *affectedObject.transportStatus* und *affectedObject.secret*:

A_16391 - (N075.300)a K_COS

Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.**[<=]**

A_16392 - (N075.300)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das Attribut *retryCounter* in einer eigenen Transaktion zeitlich vor den anderen Attributen geändert wird.**[<=]**

A_16393 - (N075.300)c K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das Attribut *secret* in einer eigenen Transaktion zeitlich vor *transportStatus* geändert wird.**[<=]**

A_16394 - (N075.300)d K_COS
Das Attribut *transportStatus* DARF NICHT persistent geändert sein, wenn nicht auch *secret* persistent geändert ist.**[<=]**

14.6.2 DISABLE VERIFICATION REQUIREMENT

Das Kommando DISABLE VERIFICATION REQUIREMENT ändert das Attribut *flagEnabled* eines Passwortobjektes (siehe 8.4 und 8.5) so, dass das COS sich so verhält, als sei der Sicherheitszustand des Passwortes ständig gesetzt. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz die als Parameter in der Kommandonachricht enthalten ist. Dieses Kommando gibt es in den Varianten mit und ohne Verifikationsdaten.

14.6.2.1 Use Case Abschalten der Benutzerverifikation mit Benutzergeheimnis

In dieser Variante wird der Zwang zur Benutzerverifikation nicht abgeschaltet, wenn das übergebene Benutzergeheimnis falsch ist. Die APDU des DISABLE VERIFICATION REQUIREMENT-Kommandos enthält zwei Parameter:

A_16396 - (N075.380) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16397 - (N075.382) K_externeWelt {K_Karte}
Der Parameter *verificationData* enthält das Benutzergeheimnis.**[<=]**

A_16398 - (N075.384) K_externeWelt {K_Karte}
Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.**[<=]**

A_16399 - (N075.386) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_b4f verwendet werden.

Tabelle 209: CosT_b4f: DISABLE VERIFICATION REQUIREMENT mit Verifikationsdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'26'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Verifikationsdaten im Datenfeld
P2	'XX'	*passwordReference*
Data	'XX…XX'	*verificationData*

[<=]

14.6.2.2 Use Case Abschalten der Benutzerverifikation ohne Benutzergeheimnis

In dieser Variante enthält die APDU des DISABLE VERIFICATION REQUIREMENT-Kommandos einen Parameter:

A_16400 - (N075.400) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16401 - (N075.500) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_a7f verwendet werden.

Tabelle 210: CosT_a7f: DISABLE VERIFICATION REQUIREMENT ohne Verifikationsdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'26'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'01'	Keine Verifikationsdaten im Datenfeld
P2	'XX'	*passwordReference*

[<=]

14.6.2.3 Antwort der Karte auf Abschalten der Benutzerverifikation

Tabelle 211: CosT_b71: DISABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	*verificationData* ist falsch
'90 00'	NoError	Erfolgreiches Abschalten des Passwortobjektes

Tabelle 212: CosT_255: DISABLE VERIFICATION REQUIREMENT Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	PasswordBlocked	Abgelaufener Fehlbedienungszähler
'69 85'	PasswordNotUsable	Passwort mit Transportschutz versehen
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_d77: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16402 - (N075.600) K_TST

Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das DISABLE VERIFICATION REQUIREMENT-Kommando zusätzliche Trailer verwendet.**[<=]**

14.6.2.4 Kommandoabarbeitung innerhalb der Karte

G2_N075.700.a - (N075.700)a K_COS
Das COS MUSS die DISABLE VERIFICATION REQUIREMENT-Variante aus
- 14.6.2.1- Use Case Abschalten der Benutzerverifikation mit Benutzergeheimnis und
- 14.6.2.2- Use Case Abschalten der Benutzerverifikation ohne Benutzergeheimnis

unterstützen.**[<=]**

A_16403 - (N075.700)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere DISABLE VERIFICATION REQUIREMENT-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N075.800 - (N075.800) K_COS
Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N075.900 - (N075.900) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

(N076.000) Diese Anforderung ist absichtlich leer.

A_16404 - (N076.100)a K_COS
Wenn der Parameter P1 den Wert '00' besitzt, dann MÜSSEN folgende Schritte abgearbeitet werden:**[<=]**

G2_N076.100.a.1 - (N076.100)a.1 K_COS
Schritt 1: Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.**[<=]**

G2_N076.100.a.2 - (N076.100)a.2 K_COS
Schritt 2: Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.**[<=]**

A_16405 - (N076.100)a.3 K_COS
Schritt 3: Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.**[<=]**

G2_N076.100.a.3.i - (N076.100)a.3.i K_COS
Wenn der Vergleich fehlschlägt, genau dann MUSS
A. *affectedObject.retryCounter* um eins dekrementiert werden und
B. clearPasswordStatus(*affectedObject*) ausgeführt werden und
C. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert 'F' gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*. **[<=]**

A_16406 - (N076.100)a.3.ii K_COS
Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.**[<=]**

G2_N076.100.a.3.iii - (N076.100)a.3.iii K_COS
Wenn der Vergleich erfolgreich ist, genau dann MUSS
A. das Attribut *affectedObject.flagEnabled* auf den Wert False gesetzt werden und
B. das Attribut *affectedObject. retryCounter* auf den Wert *affectedObject. startRetryCounter* gesetzt werden.**[<=]**

G2_N076.100.b - (N076.100)b K_COS
Falls der Parameter P1 den Wert '01'besitzt, dann MUSS das Attribut *affectedObject.flagEnabled* auf den Wert False gesetzt werden.**[<=]**

G2_N076.110 - (N076.110) K_COS
Alle persistenten Änderungen in (N076.100) MÜSSEN mit Transaktionsschutz ausgeführt werden.**[<=]**

Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für konditionale Änderung von *affectedObject. retryCounter* und die Änderung von *affectedObject. flagEnabled*:

A_16407 - (N076.120)a K_COS
Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.**[<=]**

A_16408 - (N076.120)b K_COS
Wenn das COS das Attribut *retryCounter* in einer eigenen Transaktion ändert, dann MUSS dies zeitlich vor den anderen Attributen passieren.**[<=]**

(N076.200) Diese Anforderung ist absichtlich leer.

A_16409 - (N076.300) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

(N076.400) Diese Anforderung ist absichtlich leer.

G2_N076.500.a - (N076.500)a K_COS
Wenn *affectedObject.flagEnabled* den Wert False besitzt, genau dann MUSS als Trailer NoError verwendet werden.**[<=]**

A_16410 - (N076.500)b K_COS
Wenn *affectedObject.flagEnabled* den Wert True besitzt, dann MUSS PasswordBlocked eine höhere Priorität als PasswordNotUsable haben. Die Priorität der übrigen Trailer in CosT_255 ist herstellerspezifisch.**[<=]**

G2_N076.500.c - (N076.500)c K_COS
Jeder Trailer in CosT_255 MUSS eine höhere Priorität als WrongSecretWarning haben.**[<=]**

14.6.3 ENABLE VERIFICATION REQUIREMENT

Das Kommando ENABLE VERIFICATION REQUIREMENT ändert das Attribut *flagEnabled* eines Passwortobjektes (siehe 8.4 und 8.5) so, dass der Sicherheitszustand des Passwortes nur durch eine erfolgreiche Benutzerverifikation gesetzt wird. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist. Dieses Kommando gibt es in den Varianten mit und ohne Verifikationsdaten.

14.6.3.1 Use Case Einschalten der Benutzerverifikation mit Benutzergeheimnis

In dieser Variante wird der Zwang zur Benutzerverifikation nicht eingeschaltet, falls das Benutzergeheimnis falsch ist. Die APDU des ENABLE VERIFICATION REQUIREMENT-Kommandos enthält zwei Parameter:

A_16414 - (N076.580) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16415 - (N076.582) K_externeWelt {K_Karte}
Der Parameter *verificationData* enthält das Benutzergeheimnis.**[<=]**

A_16416 - (N076.584) K_externeWelt {K_Karte}
Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.**[<=]**

A_16417 - (N076.586) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_887 verwendet werden.

Tabelle 213: CosT_887: ENABLE VERIFICATION REQUIREMENT mit Verifikationsdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'28'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Verifikationsdaten
P2	'XX'	*passwordReference*
Data	'XX…XX'	*verificationData*

[<=]

14.6.3.2 Use Case Einschalten der Benutzerverifikation ohne Benutzergeheimnis

In dieser Variante enthält die APDU des ENABLE VERIFICATION REQUIREMENT-Kommandos einen Parameter:

A_16418 - (N076.600) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16419 - (N076.700) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_44c verwendet werden.

Tabelle 214: CosT_44c: ENABLE VERIFICATION REQUIREMENT ohne Verifikationsdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'28'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'01'	Keine Verifikationsdaten
P2	'XX'	*passwordReference*

[<=]

14.6.3.3 Antwort der Karte auf Einschalten der Benutzerverifikation

Tabelle 215: CosT_2a1: ENABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	*verificationData* ist falsch
'90 00'	NoError	Erfolgreiches Einschalten des Passwortobjektes

Tabelle 216: CosT_903: ENABLE VERIFICATION REQUIREMENT Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	PasswordBlocked	Abgelaufener Fehlbedienungszähler
'69 85'	PasswordNotUsable	Passwort mit Transportschutz versehen
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_7c7: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16420 - (N076.800) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das ENABLE VERIFICATION REQUIREMENT-Kommando zusätzliche Trailer verwendet.**[<=]**

14.6.3.4 Kommandoabarbeitung innerhalb der Karte

G2_N076.900.a - (N076.900)a K_COS
Das COS MUSS die ENABLE VERIFICATION REQUIREMENT-Variante aus
- 14.6.3.1- Use Case Einschalten der Benutzerverifikation mit Benutzergeheimnis und
- 14.6.3.2- Use Case Einschalten der Benutzerverifikation ohne Benutzergeheimnis

unterstützen.**[<=]**

A_16421 - (N076.900)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere ENABLE VERIFICATION REQUIREMENT-Varianten

1. unterstützt oder
2. ablehnt.**[<=]**

G2_N077.000 - (N077.000) K_COS
Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N077.100 - (N077.100) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

(N077.200) Diese Anforderung ist absichtlich leer.

A_16422 - (N077.300)a K_COS
Wenn der Parameter P1 den Wert '00' besitzt, dann MÜSSEN folgende Schritte abgearbeitet werden:**[<=]**

G2_N077.300.a.1 - (N077.300)a.1 K_COS
Schritt 1: Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.**[<=]**

G2_N077.300.a.2 - (N077.300)a.2 K_COS
Schritt 2: Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.**[<=]**

A_16423 - (N077.300)a.3 K_COS
Schritt 3: Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.**[<=]**

G2_N077.300.a.3.i - (N077.300)a.3.i K_COS
Wenn der Vergleich fehlschlägt, genau dann MUSS
A. *affectedObject.retryCounter* um eins dekrementiert werden und
B. clearPasswordStatus(*affectedObject*) ausgeführt werden und
C. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert 'F' gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*. **[<=]**

A_16424 - (N077.300)a.3.ii K_COS
Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.**[<=]**

G2_N077.300.a.3.iii - (N077.300)a.3.iii K_COS
Wenn der Vergleich erfolgreich ist, genau dann MUSS
A. das Attribut *affectedObject.flagEnabled* auf den Wert True gesetzt werden und
B. das Attribut *affectedObject*. *retryCounter* auf den Wert *affectedObject*. *startRetryCounter* gesetzt werden.**[<=]**

G2_N077.300.b - (N077.300)b K_COS
Falls der Parameter P1 den Wert '01' besitzt, dann MUSS das Attribut *affectedObject.flagEnabled* auf den Wert True gesetzt werden.**[<=]**

G2_N077.310 - (N077.310) K_COS
Alle persistenten Änderungen in (N077.300) MÜSSEN mit Transaktionsschutz ausgeführt werden.**[<=]**

Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für konditionale Änderung von *affectedObject. retryCounter* und die Änderung von *affectedObject. flagEnabled*:

A_16425 - (N077.320)a K_COS
Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.**[<=]**

A_16426 - (N077.320)b K_COS
Wenn das COS das Attribut *retryCounter* in einer eigenen Transaktion ändert, dann MUSS dies zeitlich vor den anderen Attributen passieren.**[<=]**

(N077.400) Diese Anforderung ist absichtlich leer.

A_16427 - (N077.500) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

(N077.600) Diese Anforderung ist absichtlich leer.

G2_N077.700.a - (N077.700)a K_COS
Wenn *affectedObject.flagEnabled* den Wert True besitzt, genau dann MUSS als Trailer NoError verwendet werden.**[<=]**

A_16428 - (N077.700)b K_COS
Wenn *affectedObject.flagEnabled* den Wert False besitzt, dann MUSS PasswordBlocked eine höhere Priorität als PasswordNotUsable haben. Die Priorität der übrigen Trailer in CosT_903 ist herstellerspezifisch.**[<=]**

G2_N077.700.c - (N077.700)c K_COS
Jeder Trailer in CosT_903 MUSS eine höhere Priorität als WrongSecretWarning haben.**[<=]**

14.6.4 GET PIN STATUS

Hinweis CosH_470: Dieses Kommando ist nicht in der Normenreihe ISO/IEC 7816 enthalten. Es ließe sich kombinieren mit dem Kommando VERIFY (leere Kommandodaten). Einem DIN NIA17.4 Votum gemäß wurde auf eine derartige Kombination verzichtet.

Das Kommando GET PIN STATUS zeigt in den Antwortdaten an,

- ob der Sicherheitszustand des Passwortobjektes gesetzt ist.
- welchen Wert das Attribut *retryCounter* besitzt.
- ob ein Passwortobjekt mit einem Transportschutz versehen ist und falls ja, welches Transportschutzverfahren vom Passwortobjekt verwendet wird.
- ob eine Benutzerverifikation zum Setzen des Sicherheitsstatus erforderlich ist.

Welches Passwortobjekt von diesem Kommando betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist.

14.6.4.1 Use Case Auslesen des Status eines Passwortobjektes

In dieser Variante enthält die APDU des GET PIN STATUS-Kommandos einen Parameter:

A_16429 - (N077.800) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16430 - (N077.900) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_891 verwendet werden.

Tabelle 217: CosT_891: GET PIN STATUS

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4], hier wird "proprietary" angezeigt
INS	'20'	Instruction Byte (dies ist derselbe Wert wie beim Kommando VERIFY)
P1	'00'	–
P2	'XX'	*passwordReference*

[<=]

14.6.4.2 Antwort der Karte auf Auslesen des PIN Status

Tabelle 218: CosT_a97: GET PIN STATUS Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'62 Cx': '62 C1' '62 C7'	TransportStatus Transport–PIN Leer–PIN	Passwortobjekt eingeschaltet, Passwortobjekt ist mit Transportschutz versehen, Transportschutzverfahren enthalten im Least Significant Nibble (siehe auch 8.2.5)
'62 D0'	PasswordDisabled	Passwortobjekt ausgeschaltet, Verifikation nicht erforderlich
'63 Cx'	RetryCounter, x ungleich null	Passwortobjekt eingeschaltet, Passwortobjekt ohne Transportschutz (das bedeutet regularPassword), Wert des Fehlbedienungszählers enthalten im Least Significant Nibble
'63 C0'	RetryCounter, x gleich null	Passwortobjekt eingeschaltet, Passwortobjekt wegen Wert des Fehlbedienungszählers blockiert, Blockade ist möglicherweise mittels RESET RETRY COUNTER aufhebbar
'90 00'	NoError	Passwortobjekt eingeschaltet, Sicherheitszustand gesetzt

Tabelle 219: CosT_973: GET PIN STATUS Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
~~'63 C0'~~	~~'63 Cx' = RetryCounter, x gleich null~~	~~Passwortobjekt eingeschaltet, Passwortobjekt wegen Wert des Fehlbedienungszählers blockiert~~
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_971: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16431 - (N078.000) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das GET PIN STATUS-Kommando zusätzliche Trailer verwendet.**[<=]**

14.6.4.3 Kommandoabarbeitung innerhalb der Karte

G2_N078.100.a - (N078.100)a K_COS
Das COS MUSS die GET PIN STATUS-Variante aus
- 14.6.4.1- Use Case Auslesen des Status eines Passwortobjektes
unterstützen.**[<=]**

A_16432 - (N078.100)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere GET PIN STATUS-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N078.200 - (N078.200) K_COS
Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N078.300 - (N078.300) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_18043 - (N078.350) K_COS
Wenn *affectedObject.retryCounter* den Wert null hat, genau dann MUSS das Kommando mit dem Trailer '63 C0' terminieren.**[<=]**

G2_N078.400 - (N078.400) K_COS
Wenn das Attribut *affectedObject.flagEnabled* (siehe (N015.700)) den Wert False besitzt, genau dann MUSS als Trailer PasswordDisabled verwendet werden.**[<=]**

G2_N078.500 - (N078.500) K_COS
Wenn *affectedObject.flagEnabled* (siehe (N015.700)) den Wert True besitzt und *affectedObject* in *globalPasswordList* (siehe (N029.900)i) oder in *dfSpecificPasswordList* (siehe (N029.900)j) enthalten ist und dort das Attribut *securityStatusEvaluationCounter*

(siehe (N029.900)k) einen Wert ungleich null besitzt, genau dann MUSS als Trailer NoError verwendet werden.**[<=]**

G2_N078.600 - (N078.600) K_COS
Wenn das Attribut *affectedObject.transportStatus* (siehe (N015.600)) einen Wert ungleich regularPassword besitzt, genau dann MUSS als Trailer TransportStatus verwendet werden mit TransportStatus = '62 Cx'. Dabei ist 'x' durch die Codierung von*affectedObject.transportStatus* gemäß CosT_a97 zu ersetzen.**[<=]**

G2_N078.700 - (N078.700) K_COS
Wenn das Attribut *affectedObject.transportStatus* (siehe (N015.600)) den Wert regularPassword besitzt, genau dann MUSS als Trailer RetryCounter verwendet werden mit RetryCounter = '63 Cx'. Dabei ist 'x' zu ersetzen durch das Minimum der beiden Zahlen 15 = 'F' und*affectedObject.retryCounter*. **[<=]**

A_16433 - (N078.800)a K_TST
Die Priorität der Trailer in CosT_973 MUSS herstellerspezifisch sein.**[<=]**

G2_N078.800.b - (N078.800)b.1 K_COS
Jeder Trailer in CosT_973 MUSS eine höhere Priorität als RetryCounter = '63 C0" haben.**[<=]**

A_18353 - (N078.800)b.2 K_COS
RetryCounter = '63 C0' MUSS eine höhere Priorität als PasswordDisabled haben.**[<=]**

G2_N078.800.c - (N078.800)c K_COS
PasswordDisabled MUSS eine höhere Priorität als NoError haben.**[<=]**

G2_N078.800.d - (N078.800)d K_COS
NoError MUSS eine höhere Priorität als TransportStatus haben.**[<=]**

G2_N078.800.e - (N078.800)e K_COS
TransportStatus MUSS eine höhere Priorität als RetryCounter= '63 Cx' mit x ungleich null haben.**[<=]**

Hinweis CosH_73f: Gemäß (N078.600), (N078.700) und (N078.800) ist es nicht möglich den Fehlbedienungszähler auszulesen, wenn Transportschutz besteht. Gemäß (N078.350) und (N078.800) wird ein blockiertes Passwortobjekt auch bei bestehendem Transportschutz erkannt.

14.6.5 RESET RETRY COUNTER

Das Kommando RESET RETRY COUNTER setzt das Attribut *retryCounter* eines Passwortobjektes auf den Startwert *startRetryCounter*. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist. Dieses Kommando gibt es in den Varianten

- mit *PUK*, mit neuem Wert für das Attribut *secret* des Passwortobjektes
- mit *PUK*, ohne neuem Wert für das Attribut *secret* des Passwortobjektes
- ohne *PUK*, mit neuem Wert für das Attribut *secret* des Passwortobjektes
- ohne *PUK*, ohne neuem Wert für das Attribut *secret* des Passwortobjektes

Die Variante "mit *PUK*" wird typischerweise verwendet, wenn eine Person (etwa der Karteninhaber) das Recht zur Durchführung dieser Aktion besitzt.

Die Variante "ohne *PUK*" wird typischerweise verwendet, wenn eine technische Instanz (etwa das CMS) diese Aktion durchführt. In den Zugriffsbedingungen wird dann typischerweise zumindest eine Rollenauthentisierung gefordert.

Die Variante "mit neuem Geheimnis" wird typischerweise gewählt, wenn der Karteninhaber sein Passwort definitiv vergessen hat.

Die Variante "ohne neues Geheimnis" wird typischerweise gewählt, wenn es der Instanz, welche zum Rücksetzen des Fehlbedienungszählers berechtigt ist, verboten ist, durch das Passwort geschützte Aktionen auszuführen. Dies ist häufig im Umfeld qualifizierter Signaturanwendungen und dem Signaturpasswort anzutreffen.

14.6.5.1 Use Case Entsperren mit PUK, mit neuem Geheimnis

In dieser Variante enthält die APDU des RESET RETRY COUNTER-Kommandos drei Parameter:

A_16444 - (N078.900) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16445 - (N079.000) K_externeWelt {K_Karte}
Der Parameter *PUK* MUSS ein Geheimnis enthalten, welches diese Aktion autorisiert.**[<=]**

A_16446 - (N079.100) K_externeWelt {K_Karte}
Der Parameter *newSecret* MUSS das neue Benutzergeheimnis enthalten.**[<=]**

A_16447 - (N079.200) K_externeWelt {K_Karte}
Die Parameter *PUK* und *newSecret* MÜSSEN gemäß (N008.100) codiert sein.**[<=]**

A_16448 - (N079.300) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_0d1 verwendet werden.

Tabelle 220: CosT_0d1: RESET RETRY COUNTER, mit PUK, mit newSecret

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Data enthält *PUK* und neues Benutzergeheimnis
P2	'XX'	*passwordReference*
Data	'XX…XX'	*PUK* || *newSecret*

[<=]

14.6.5.2 Use Case Entsperren mit PUK, ohne neues Geheimnis

In dieser Variante enthält die APDU des RESET RETRY COUNTER-Kommandos zwei Parameter:

A_16449 - (N079.400) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16450 - (N079.500) K_externeWelt {K_Karte}
Der Parameter *PUK* enthält ein Geheimnis, welches diese Aktion autorisiert.**[<=]**

A_16451 - (N079.600) K_externeWelt {K_Karte}
Der Parameter *PUK* MUSS gemäß (N008.100) codiert sein.**[<=]**

A_16452 - (N079.700) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_f19 verwendet werden.

Tabelle 221: CosT_f19: RESET RETRY COUNTER, mit PUK, ohne newSecret

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'01'	Data enthält nur *PUK*
P2	'XX'	*passwordReference*
Data	'XX…XX'	*PUK*

[<=]

14.6.5.3 Use Case Entsperren ohne PUK, mit neuem Geheimnis

In dieser Variante enthält die APDU des RESET RETRY COUNTER-Kommandos zwei Parameter:

A_16453 - (N079.800) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16454 - (N079.900) K_externeWelt {K_Karte}
Der Parameter *newSecret* enthält das neue Benutzergeheimnis.**[<=]**

A_16455 - (N080.000) K_externeWelt {K_Karte}
Der Parameter *newSecret* MUSS gemäß (N008.100) codiert sein.**[<=]**

A_16456 - (N080.100) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_a37 verwendet werden.

Tabelle 222: CosT_a37: RESET RETRY COUNTER, ohne PUK, mit newSecret

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'02'	Data enthält nur neues Benutzergeheimnis
P2	'XX'	*passwordReference*
Data	'XX…XX'	*newSecret*

[<=]

14.6.5.4 Use Case Entsperren ohne PUK, ohne neues Geheimnis

In dieser Variante enthält die APDU des RESET RETRY COUNTER-Kommandos einen Parameter:

A_16457 - (N080.200) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16458 - (N080.300) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_551 verwendet werden.

Tabelle 223: CosT_551: RESET RETRY COUNTER, ohne PUK, ohne newSecret

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2C'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'03'	Kommandodatenfeld fehlt
P2	'XX'	*passwordReference*

[<=]

14.6.5.5 Antwort der Karte auf Entsperren eines Benutzergeheimnisses

Tabelle 224: CosT_f92: RESET RETRY COUNTER Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	*PUK* ist falsch
'90 00'	NoError	Erfolgreiches Rücksetzen des Fehlbedienungszählers

Tabelle 225: CosT_a92: RESET RETRY COUNTER Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	CommandBlocked	Bedienungszähler der *PUK* abgelaufen
'69 85'	LongPassword	*newData* enthält ein zu langes Passwort
'69 85'	ShortPassword	*newData* enthält ein zu kurzes Passwort
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_775: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16459 - (N080.400) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das RESET RETRY COUNTER-Kommando zusätzliche Trailer verwendet.**[<=]**

14.6.5.6 Kommandoabarbeitung innerhalb der Karte

G2_N080.500.a - (N080.500)a K_COS
Das COS MUSS die RESET RETRY COUNTER-Varianten aus
- 14.6.5.1- Use Case Entsperren mit PUK, mit neuem Geheimnis,
- 14.6.5.2- Use Case Entsperren mit PUK, ohne neues Geheimnis,
- 14.6.5.3- Use Case Entsperren ohne PUK, mit neuem Geheimnis und
- 14.6.5.4- Use Case Entsperren ohne PUK, ohne neues Geheimnis

unterstützen.**[<=]**

A_16460 - (N080.500)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere RESET RETRY COUNTER-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N080.600 - (N080.600) K_COS
Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N080.700 - (N080.700) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N080.800.a - (N080.800)a K_COS
Wenn die konditional vorhandene und in *newSecret* codierte Ziffernfolge für das Attribut *secret* des Passwortobjektes eine Länge hat, die kleiner als *affectedObject.minimumLength* ist, genau dann MUSS das Kommando mit dem Trailer ShortPassword terminieren.**[<=]**

G2_N080.800.b - (N080.800)b K_COS
Wenn die konditional vorhandene und in *newSecret* codierte Ziffernfolge für das Attribut *secret* des Passwortobjektes eine Länge hat, die größer als *affectedObject.maximumLength* ist, genau dann MUSS das Kommando mit dem Trailer LongPassword terminieren.**[<=]**

G2_N080.900 - (N080.900) K_COS

Mittels clearPasswordStatus(*affectedObject*) MUSS der Sicherheitsstatus zurückgesetzt werden.**[<=]**

A_16461 - (N081.000) K_COS
Wenn das Kommandodatenfeld einen Parameter *PUK* enthält, genau dann MUSS dieser mit dem Attribut *affectedObject.PUK* verglichen werden.**[<=]**

G2_N081.000.a - (N081.000)a K_COS
Wenn *affectedObject.pukUsage* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer CommandBlocked terminieren.**[<=]**

A_16462 - (N081.000)b K_COS
affectedObject.pukUsage MUSS mit Transaktionsschutz um eins dekrementiert werden.**[<=]**

G2_N081.000.c - (N081.000)c K_COS
Wenn der Vergleich fehlschlägt, genau dann MUSS das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert 'F' gesetzt werden, wenn*affectedObject.pukUsage* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.pukUsage*. **[<=]**

Hinweis CosH_b46: Die Bearbeitungsreihenfolge der Punkte (N080.700), (N080.800), (N080.900) und (N081.000) ist COS spezifisch. Deshalb ist es COS spezifisch, ob in gewissen Fehlerfällen der Sicherheitsstatus von affectedObject *zurückgesetzt wird oder nicht.*

G2_N081.100 - (N081.100) K_COS
Das Attribut *affectedObject.retryCounter* MUSS auf den Wert *affectedObject.startRetryCounter* gesetzt werden.**[<=]**

A_16463 - (N081.200) K_COS
Wenn das Kommandodatenfeld einen Parameter *newSecret* enthält, genau dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N081.200.a - (N081.200)a K_COS
Schritt 1: *affectedObject.secret* MUSS auf den in *newSecret* codierten Wert gesetzt werden.**[<=]**

G2_N081.200.b - (N081.200)b K_COS
Schritt 2: *affectedObject.transportStatus* MUSS auf den Wert "regularPassword" (siehe (N009.500)) gesetzt werden, falls dieses Attribut einen anderen Wert besitzt.**[<=]**

G2_N081.300 - (N081.300) K_COS
Die persistenten Änderungen in (N081.000), (N081.100) und (N081.200) MÜSSEN mit Transaktionsschutz ausgeführt werden.**[<=]**

(N081.400) Diese Anforderung ist absichtlich leer.

A_16464 - (N081.500) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N081.600 - (N081.600) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16465 - (N081.700)a K_TST
Die Priorität der Trailer in CosT_a92 MUSS herstellerspezifisch sein.**[<=]**

G2_N081.700.b - (N081.700)b K_COS

Jeder Trailer in CosT_a92 MUSS eine höhere Priorität als WrongSecretWarning haben.**[<=]**

G2_N081.700.c - (N081.700)c K_COS
WrongSecretWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für die möglichen Änderungen von *secret*, *transportStatus* und *retryCounter*:

G2_N081.800.a - (N081.800)a K_COS
Alle Änderungen SOLLEN gemeinsam in einer Transaktion geändert werden.**[<=]**

A_16466 - (N081.800)b.1 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass die Änderungen von *secret*, *transportStatus* oder *retryCounter* in eigenen Transaktionen stattfinden.**[<=]**

G2_N081.800.b - (N081.800)b.2 K_COS
Wenn die Änderungen von *secret*, *transportStatus* oder *retryCounter* in eigenen Transaktionen stattfinden, dann MUSS *retryCounter* als letztes geändert werden.**[<=]**

14.6.6 VERIFY

Das Kommando VERIFY vergleicht das Attribut *secret* eines Passwortobjektes mit Daten, die im Datenfeld der Kommandonachricht enthalten sind. Falls der Vergleich erfolgreich ist, wird der Sicherheitszustand der Karte geändert. Welches Passwortobjekt betroffen ist, bestimmt eine Passwortreferenz, die als Parameter in der Kommandonachricht enthalten ist.

14.6.6.1 Use Case Vergleich eines Benutzergeheimnisses

In dieser Variante enthält die APDU des VERIFY Kommandos zwei Parameter:

A_16467 - (N081.900) K_externeWelt {K_Karte}
Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.**[<=]**

A_16468 - (N082.000) K_externeWelt {K_Karte}
Der Parameter *verificationData* enthält das Benutzergeheimnis.**[<=]**

A_16469 - (N082.100) K_externeWelt {K_Karte}
Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.**[<=]**

A_16470 - (N082.200) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_3b5 verwendet werden.

Tabelle 226: CosT_3b5: VERIFY, Vergleich eines Benutzergeheimnisses

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'20'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Data enthält Verifikationsdaten
P2	'XX'	*passwordReference*
Data	'XX…XX'	*verificationData*

[<=]

14.6.6.2 Antwort der Karte auf Vergleich eines Benutzergeheimnisses

Tabelle 227: CosT_148: VERIFY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	*verificationData* ist falsch
'90 00'	NoError	Erfolgreicher Vergleich

Tabelle 228: CosT_49d: VERIFY Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	PasswordBlocked	Abgelaufener Fehlbedienungszähler
'69 85'	PasswordNotUsable	Passwort mit Transportschutz versehen
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis CosH_839: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16471 - (N082.300) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das VERIFY-Kommando zusätzliche Trailer verwendet.**[<=]**

14.6.6.3 Kommandoabarbeitung innerhalb der Karte

G2_N082.400.a - (N082.400)a K_COS
Das COS MUSS die VERIFY-Variante aus
- 14.6.6.1- Use Case Vergleich eines Benutzergeheimnisses

unterstützen.**[<=]**

A_16472 - (N082.400)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere VERIFY-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N082.500 - (N082.500) K_COS
Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Wenn die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.**[<=]**

G2_N082.600 - (N082.600) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N082.700 - (N082.700) K_COS
Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.**[<=]**

G2_N082.800 - (N082.800) K_COS
Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.**[<=]**

A_16473 - (N082.900) K_COS
Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.**[<=]**

G2_N082.900.a - (N082.900)a K_COS
Wenn der Vergleich fehlschlägt, genau dann MUSS
1. *affectedObject.retryCounter* um eins dekrementiert werden und
2. clearPasswordStatus(*affectedObject*) ausgeführt werden und
3. das Kommando mit dem Trailer WrongSecretWarning terminieren. Das Lownibble des Trailers MUSS dabei auf den Wert 'F' gesetzt werden, wenn*affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*. **[<=]**

G2_N082.900.b - (N082.900)b K_COS
Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.**[<=]**

A_16474 - (N082.900)c K_COS
Wenn der Vergleich erfolgreich ist, genau dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N082.900.c.1 - (N082.900)c.1 K_COS
Schritt 1: Es MUSS setPasswordStatus(*affectedObject*) ausgeführt werden.**[<=]**

G2_N082.900.c.2 - (N082.900)c.2 K_COS
Schritt 2: Das Attribut MUSS *affectedObject.retryCounter* auf den Wert *affectedObject.startRetryCounter* gesetzt werden.**[<=]**

G2_N082.900.c.3 - (N082.900)c.3 K_COS
Die Änderung von *affectedObject.retryCounter* MUSS mit Transaktionsschutz ausgeführt werden.**[<=]**

(N083.000) Diese Anforderung ist absichtlich leer.

A_16475 - (N083.100) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N083.200 - (N083.200) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16476 - (N083.300)a K_COS
PasswordBlocked MUSS eine höhere Priorität als PasswordNotUsable haben. Die Priorität der übrigen Trailer in CosT_49d ist herstellerspezifisch.**[<=]**

G2_N083.300.b - (N083.300).b K_COS
Jeder Trailer in CosT_49d MUSS eine höhere Priorität als WrongSecretWarning haben.**[<=]**

A_16477 - (N083.300)c) K_COS
WrongSecretWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.7 Komponentenauthentisierung

14.7.1 EXTERNAL AUTHENTICATE / MUTUAL AUTHENTICATE

Die Kommandos EXTERNAL AUTHENTICATE und MUTUAL AUTHENTICATE prüfen die Authentizität einer externen Instanz anhand der Antwort auf ein von der Karte generiertes Token, mittels eines symmetrischen oder öffentlichen Schlüssels. Der Schlüssel wird vor der Authentisierungsoperation ausgewählt. Dies geschieht vor dem Senden dieses Kommandos durch ein MSE-Set-Kommando (siehe (N101.400), (N101.900), (N102.400)). Die Antwort der externen Instanz auf das von der Karte generierte Token ist als Parameter in der Kommandonachricht enthalten.

~~*Hinweis (107): In [gemSpec_eGK_P1] wurden die Varianten EXTERNAL versus MUTUAL anhand eines Algorithm Identifiers unterschieden. Hier werden die Varianten, wie schon in [HBA_P1], anhand des LeFeldes unterschieden*~~

Hinweis CosH_e15: Der Wert '82' für das INS-Byte wird für die Kommandovarianten EXTERNAL AUTHENTICATE und MUTUAL AUTHENTICATE verwendet. Die Varianten lassen sich anhand der Existenz des LeFeldes unterscheiden. Davon wird in (N084.400) und (N084.410) Gebrauch gemacht. Dies schließt die Verwendung des Protokolls T=0 für Karten in der Telematikinfrastruktur aus. Dies erscheint vertretbar, da es derzeit für die Verwendung des Protokolls T=0 keine Anforderung gibt.

14.7.1.1 Use Case Externes Authentisieren ohne Antwortdaten

In dieser Variante enthält die APDU des EXTERNAL AUTHENTICATE-Kommandos einen Parameter: Der Parameter *cmdData* enthält die Antwort der externen Instanz. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *cmdData* MUSS abhängig von der mittels (N101.300), (N101.800) oder (N102.300) ausgewählten *algId* gewählt werden.

A_16480 - (N083.400)a K_externeWelt {K_Karte}
Wenn *algId* gleich elcRoleCheck ist, dann MUSS unter Hinzuziehung der Domainparameter des öffentlichen Schlüssels gelten: 8 OctetLength(*cmdData*) = 2 *domainParameter*. *t*. **[<=]**

(N083.400)b ist absichtlich leer.
(N083.400)c ist absichtlich leer.

A_16481 - (N083.400)b K_externeWelt {K_Karte}, Option_RSA_CVC
~~Wenn *algId* gleich rsaRoleCheck ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.~~**[<=]**

A_16482 - (N083.400)c K_externeWelt {K_Karte}, Option_DES

Wenn *algId* gleich rsaSessionkey4SM ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.**[<=]**

A_16483 - (N083.402)a K_externeWelt {K_Karte}, Option_Kryptobox
Wenn *algId* gleich aesSessionkey4TC ist, dann MUSS *cmdData* gleich 120 Oktett lang sein.**[<=]**

(N083.402)b ist absichtlich leer.
(N083.402)c ist absichtlich leer.

A_16484 - (N083.402)b K_externeWelt {K_Karte}, Option_Kryptobox und Option_DES
Wenn *algId* gleich desSessionkey4TC ist, dann MUSS *cmdData* gleich 104 Oktett lang sein.**[<=]**

A_16485 - (N083.402)c K_externeWelt {K_Karte}, Option_Kryptobox und Option_DES
Wenn *algId* gleich rsaSessionkey4TC ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.**[<=]**

A_16486 - (N083.500) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_c1c verwendet werden.

Tabelle 229: CosT_c1c: EXTERNAL AUTHENTICATE ohne Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'82'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Information zum Algorithmus bereits in der Karte vorhanden
P2	'00'	Schlüsselreferenz bereits in der Karte vorhanden
Data	'XX…XX'	*cmdData*

[<=]

14.7.1.2 Use Case Externes Authentisieren mit Antwortdaten

In dieser Variante enthält die APDU des MUTUAL AUTHENTICATE-Kommandos zwei Parameter:

A_16491 - (N083.600) K_externeWelt {K_Karte}
Der Parameter *cmdData* enthält die Antwort der externen Instanz. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *cmdData* MUSS abhängig von der mittels (N102.300) ausgewählten *algId* gewählt werden.**[<=]**

A_16492 - (N083.600)a K_externeWelt {K_Karte}
Wenn *algId* gleich aesSessionkey4SM ist, dann MUSS *cmdData* 120 Oktett lang sein.**[<=]**

(N083.600)b ist absichtlich leer.

A_16493 - (N083.600)b K_externeWelt {K_Karte}, Option_DES
Wenn *algId* gleich desSessionkey4SM ist, dann MUSS *cmdData* 104 Oktett lang sein.**[<=]**

A_16494 - (N083.700) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.**[<=]**

A_16495 - (N083.800) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_fe0 verwendet werden.

Tabelle 230: CosT_fe0: MUTUAL AUTHENTICATE mit Antwortdaten

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'82'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Information zum Algorithmus bereits in der Karte vorhanden
P2	'00'	Schlüsselreferenz bereits in der Karte vorhanden
Data	'XX…XX'	*cmdData*
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.1.3 Antwort der Karte auf Externes Authentisieren

Tabelle 231: CosT_1fd: EXTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*authData*	Authentisierungsdaten (konditional, abhängig von *algId*)
Trailer	**Inhalt**	**Beschreibung**
'63 00'	AuthenticationFailure	Authentisierung fehlgeschlagen
'90 00'	NoError	Erfolgreiche Authentisierung

Tabelle 232: CosT_e88: EXTERNAL AUTHENTICATE Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	KeyExpired	Gültigkeitszeitraum des Schlüssels ist abgelaufen
'69 85'	NoKeyReference	Kein Authentisierungsschlüssel ausgewählt
~~'69 85'~~	~~NoPrkReference~~	~~Kein Entschlüsselungsschlüssel ausgewählt~~
'69 85'	NoRandom	Keine Zufallszahl vorhanden (siehe auch 15.1)
'69 85'	WrongRandomLength	Zufallszahl hat die falsche Länge
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 88'	KeyNotFound	Authentisierungsschlüssel nicht gefunden
~~'6A 88'~~	~~PrKNotFound~~	~~Entschlüsselungsschlüssel nicht gefunden~~

Hinweis CosH_34e: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16496 - (N083.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das EXTERNAL AUTHENTICATE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.7.1.4 Kommandoabarbeitung innerhalb der Karte

G2_N084.000.a - (N084.000)a K_COS
Das COS MUSS die EXTERNAL AUTHENTICATE-Variante aus
- 14.7.1.1- Use Case Externes Authentisieren ohne Antwortdaten und
- 14.7.1.2- Use Case Externes Authentisieren mit Antwortdaten

unterstützen.**[<=]**

G2_N084.000.b - (N084.000)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere EXTERNAL AUTHENTICATE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N084.100 - (N084.100) K_COS
Wenn *RND.ICC* den Wert "NoRandom" besitzt (siehe (N099.300), (N029.900)b undKapitel 15), genau dann MUSS das Kommando mit dem Trailer NoRandom terminieren.**[<=]**

G2_N084.200.a - (N084.200)a K_COS
Wenn *channelContext.keyReferenceList.externalAuthenticate* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16497 - (N084.200)b K_COS
Wenn *channelContext.keyReferenceList.externalAuthenticate* nicht leer ist, dann wird *affectedObject* = SearchKey(
channelContext. currentFolder
keyReferenceList.externalAuthenticate.keyReference,
keyReferenceList.externalAuthenticate.algID

) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler
1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N084.220.a - (N084.220)a K_COS
Das Kommando MUSS mit dem Trailer WrongRandomLength terminieren, falls *RND.ICC* eine Länge ungleich 16 ~~von acht~~ Oktette besitzt und *keyReferenceList.externalAuthenticate.algID* Element der folgenden Menge ist: {aesSessionkey4*, elcRoleCheck}.**[<=]**

(N084.220)b ist absichtlich leer.

G2_N084.220.b - (N084.220)b K_COS, Option_DES, Option_RSA_CVC
~~Das Kommando MUSS mit dem Trailer WrongRandomLength terminieren, falls *RND.ICC* eine Länge von 16 Oktetten besitzt und *keyReferenceList.externalAuthenticate.algID* Element der folgenden Menge ist: {~~
~~desSessionkey4* (Option_DES),~~
~~rsa* (Option_RSA_CVC)~~
~~}.~~**[<=]**

A_16498 - (N084.280) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass der Sicherheitszustand mittels clearSecurityStatusKey(*affectedObject*) vor der Auswertung der Zugriffsregel (siehe (N084.300)) oder vor der Auswertung der Kommandodaten (siehe (N084.400)) zurückgesetzt wird. In diesem Fall ist das Löschen des Sicherheitszustandes gemäß (N084.500) nicht erforderlich.**[<=]**

G2_N084.300 - (N084.300) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_16499 - (N084.400) K_COS
Die Antwort der externen Instanz ist in *cmdData* enthalten. Wenn die Kommando APDU kein LeFeld enthält, mithin also ein Case 3 gemäß 11.7.3 vorliegt, dann handelt es sich um die EXTERNAL AUTHENTICATE-Variante des Kommandos und es gilt *authData* = '' (leerer Oktettstring). In diesem Fall MUSS das COS *cmdData* wie folgt verarbeiten:**[<=]**

A_16500 - (N084.400)a K_COS
Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert elcRoleCheck besitzt, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N084.400.a.1 - (N084.400)a.1 K_COS
Schritt 1: Wenn *affectedObject. expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), genau dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.**[<=]**

G2_N084.400.a.2 - (N084.400)a.2 K_COS
Schritt 2: *cmdData* MUSS wie folgt auf R und S aufgeteilt werden:
i. *cmdData* = R || S.
ii. OctetLength(R) == OctetLength(S).**[<=]**

G2_N084.400.a.3 - (N084.400)a.3 K_COS

Schritt 3: Setze *PuK* = *affectedObject.publicKey*. **[<=]**

G2_N084.400.a.4 - (N084.400)a.4 K_COS
Schritt 4: Setze *hash* = I2OS(OS2I(*RND.ICC* || *iccsn8* || elcRoleCheck), *PuK. domainParameter*. *t* / 8).
Hinweis CosH_438: iccsn8 *ist in (N019.900)c definiert.***[<=]**

G2_N084.400.a.5 - (N084.400)a.5 K_COS
Schritt 5: Setze *out* = ELC_VER_SIG(*PuK*, *R*, *S*, *hash*).
Wenn *out* den Wert *False* besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.**[<=]**

(N084.400)b ist absichtlich leer.
(N084.400)c ist absichtlich leer.

A_16501 - (N084.400)b K_COS, Option_RSA_CVC
Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert rsaRoleCheck besitzt, dann MÜSSEN mit der Definition *PuK = affectedObject.publicKey* folgende Schritte ausgeführt werden:**[<=]**

G2_N084.400.b.1 - (N084.400)b.1 K_COS, Option_RSA_CVC
Schritt 1: Setze *M2 = RND.ICC || iccsn8.*
Hinweis CosH_cdd: iccsn8 *ist in (N019.900)c definiert.***[<=]**

G2_N084.400.b.2 - (N084.400)b.2 K_COS, Option_RSA_CVC
Schritt 2: Setze (*out*, *M*) = RSA_ISO9796_2_DS1_VERIFY(*PuK*, *cmdData*, *M2*).
Wenn diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.**[<=]**

A_16502 - (N084.400)c K_COS, Option_DES
Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert rsaSessionkey4SM besitzt, dann gilt *authData* = '' (leerer Oktettstring) und es MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N084.400.c.1.i - (N084.400Kc.1.i K_COS, Option_DES
Schritt 1.i: Wenn *channelContext.keyReferenceList.internalAuthenticate* leer ist, genau dann MUSS das Kommando mit dem Trailer NoPrkReference terminieren.**[<=]**

A_16503 - (N084.400)c.1.ii K_COS, Option_DES
Schritt 1.ii: Wenn *channelContext.keyReferenceList.internalAuthenticate* nicht leer ist, dann MUSS
mpObject = SearchKey(
channelContext. currentFolder,
keyReferenceList.internalAuthenticate.keyReference,
keyReferenceList.externalAuthenticate.algID
) gesetzt werden. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N084.400.c.1.ii.A - (N084.400)c.1.ii.A K_COS, Option_DES
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.**[<=]**

G2_N084.400.c.1.ii.B - (N084.400)c.1.ii.B K_COS, Option_DES
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

A_16504 - (N084.400)c.2 K_TST, Option_DES

~~Schritt 2: Die funktionale Eignung DARF einem Prüfling NICHT mit der Begründung verwehrt werden für den privaten Schlüssel in *tmpObject* würde gemäß eine Zugriffsregelprüfung für AccessRuleEvaluation(*tmpObject*, *CLA*, *INS*, *P1*, *P2*) eine Prüfung der Zugriffsregeln stattfinden oder unterbleiben (siehe CosH_2bc).~~**[<=]**

G2_N084.400.c.3 - (N084.400)c.3 K_COS, Option_DES
~~Schritt 3 :Setze *tmpData = cmdData*$^{PrK.d}$ mod *PrK.n*, mit *PrK = tmpObject.privateKey*.~~**[<=]**

G2_N084.400.c.4 - (N084.400)c.4 K_COS, Option_DES
~~Schritt 4: Setze *M2 = RND.ICC || iccsn8*.~~**[<=]**

G2_N084.400.c.5 - (N084.400)c.5 K_COS, Option_DES
~~Schritt 5: Setze (*out*, *M*) = RSA_ISO9796_2_DS1_VERIFY(*PuK*, *tmpData*, *M2*). Wenn diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.~~**[<=]**

G2_N084.400.c.6 - (N084.400)c.6 K_COS, Option_DES
~~Schritt 6: *M* wird aufgeteilt in *M = M1 || M2*. Die 64 LSByte in *M1* MÜSSEN als KD.e an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).~~**[<=]**

A_16505 - (N084.402) K_COS, Option_Kryptobox
Die Antwort der externen Instanz ist in *cmdData* enthalten. Falls die Kommando APDU kein LeFeld enthält, mithin also ein Case 3 gemäß 11.7.3 vorliegt, dann handelt es sich um die EXTERNAL AUTHENTICATE-Variante des Kommandos und es gilt *authData* = '' (leerer Oktettstring). In diesem Fall MUSS *cmdData* wie folgt weiterverarbeitet werden:**[<=]**

G2_N084.402.a - (N084.402)a K_COS, Option_Kryptobox
Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert aesSessionkey4TC besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:

1. *cmdData* wird wie folgt aufgeteilt:
   i. *cmdData* == *C1* || *MAC1*
   ii. OctetLength(*MAC1*) == 8.
2. *out* = VerifyCMAC_IsoPadding(*affectedObject.macKey*, *MAC1*, *C1*)
3. Wenn *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
4. *R* = AES_CBC_DEC(*affectedObject.encKey*, 0, *C1*)
5. *R* wird wie folgt aufgeteilt:
   i. *R* == RND.ext || ICCSN8.ext || RND.int || ICCSN8.int || KD.e
   ii. Es sei 16 == OctetLength(RND.ext)
   iii. Es sei 8 == OctetLength(ICCSN8.ext)
   iv. Es sei 16 == OctetLength(RND.int)
   v. Es sei 8 == OctetLength(ICCSN8.int)
   vi. Es sei 64 == OctetLength(KD.e)
6. Das Kommando MUSS mit AuthenticationFailure terminieren, wenn
   i. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
   ii. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist, oder
   iii. ICCSN8.int identisch zu ICCSN8.ext ist.
7. KD.e MUSS an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).

[<=]

(N084.402)b ist absichtlich leer.
(N084.402)c ist absichtlich leer.

G2_N084.402.b - (N084.402)b K_COS, Option_Kryptobox, Option_DES
Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert desSessionkey4TC besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:

1. *cmdData* wird wie folgt aufgeteilt:
   i. *cmdData* = *C1* || *MAC1*
   ii. OctetLength(*MAC1*) = 8.
2. *out* = VERIFY_Retail_MAC(*affectedObject.macKey*, *MAC1*, *C1*)
3. Wenn *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
4. *R* = 3TDES_CBC_DEC(*affectedObject.encKey*, 0, *C1*)
5. *R* wird wie folgt aufgeteilt:
   i. *R* = RND.ext || ICCSN8.ext || RND.int || ICCSN8.int || KD.e
   ii. Es sei 8 = OctetLength(RND.ext)
   iii. Es sei 8 = OctetLength(ICCSN8.ext)
   iv. Es sei 8 = OctetLength(RND.int)
   v. Es sei 8 = OctetLength(ICCSN8.int)
   vi. Es sei 64 = OctetLength(KD.e)
6. Das Kommando MUSS mit AuthenticationFailure terminieren, wenn
   i. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
   ii. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist, oder
   iii. ICCSN8.int identisch zu ICCSN8.ext ist.
7. KD.e MUSS an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).

[<=]

G2_N084.402.c - (N084.402)c K_COS, Option_Kryptobox, Option_DES
Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert rsaSessionkey4TC besitzt, dann MUSS im Rahmen der Kommandobearbeitung identisch zu rsaSessionkey4SM verfahren werden.**[<=]**

A_16506 - (N084.410) K_COS
Die Antwort der externen Instanz ist in *cmdData* enthalten. Falls die Kommando APDU ein LeFeld enthält, mithin also ein Case 4S gemäß 11.7.4.1 vorliegt, dann handelt es sich um die MUTUAL AUTHENTICATE-Variante des Kommandos. In diesem Fall MUSS *cmdData* wie folgt weiterverarbeitet werden, wobei *authData* berechnet wird: **[<=]**

G2_N084.410.a - (N084.410)a K_COS
Wenn *channelContext.keyReferenceList.externalAuthenticate.algID* den Wert aesSessionkey4SM besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:

1. *cmdData* wird wie folgt aufgeteilt:
2. *cmdData* == *C1* || *MAC1*

3. OctetLength(*MAC1*) == 8
4. *out* = VerifyCMAC_IsoPadding(*affectedObject.macKey*, *MAC1*, *C1*)
5. Wenn *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren, sonst MÜSSEN die folgenden Schritte durchgeführt werden:
6. *S* = AES_CBC_DEC(*affectedObject.encKey*, 0, *C1*)
   i. *S* wird wie folgt aufgeteilt:
   ii. *S* == RND.ext || ICCSN8.ext || RND.int || ICCSN8.int || KD.e
   iii. Es sei 16 == OctetLength(RND.ext)
   iv. Es sei 8 == OctetLength(ICCSN8.ext)
   v. Es sei 16 == OctetLength(RND.int)
   vi. Es sei 8 == OctetLength(ICCSN8.int)
   vi.. Es sei 64 == OctetLength(KD.e)
7. Das Kommando MUSS mit AuthenticationFailure terminieren, wenn
   i. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
   ii. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist.
   iii. ICCSN8.int identisch zu ICCSN8.ext ist.
8. Setze KD.i = RAND(64)
9. Setze *R* = RND.int || ICCSN8.int || RND.ext || ICCSN8.ext || KD.i
10. Setze *C2* = AES_CBC_ENC(*affectedObject.encKey*, 0, *R*)
11. Setze *MAC2* = CalculateCMAC_IsoPadding(*affectedObject.macKey*, *C2*)
12. Setze *authData* = *C2* || *MAC2*
13. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).

[<=]

(N084.410)b ist absichtlich leer.

G2_N084.410.b - (N084.410)b K_COS, Option_DES
~~Wenn *channelContext.keyReferenceList.externalAuthenticate.algID* den Wert desSessionkey4SM besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:~~

1. ~~*cmdData* wird wie folgt aufgeteilt:~~
2. ~~*cmdData* = *C1* || *MAC1*~~
3. ~~OctetLength(*MAC1*) = 8.~~
4. ~~*out* = VERIFY_Retail_MAC(*affectedObject.macKey*, *MAC1*, *C1*). Wenn *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.~~
5. ~~Der Oktettstring *C1* in den Kommandodaten wird wie folgt behandelt:~~
   ~~i. *S* = 3TDES_CBC_DEC(*affectedObject.encKey*, 0, *C1*)~~
   ~~ii. *S* wird wie folgt aufgeteilt:~~
   ~~iii. *S* = RND.ext || ICCSN8.ext || RND.int || ICCSN8.int || KD.e~~

iv. Es sei 8 = OctetLength(RND.ext)
v. Es sei 8 = OctetLength(ICCSN8.ext)
vi. Es sei 8 = OctetLength(RND.int)
vii. Es sei 8 = OctetLength(ICCSN8.int)
viii. Es sei 64 = OctetLength(KD.e)

6. Das Kommando MUSS mit AuthenticationFailure terminieren, wenn
   i. RND.int ungleich *RND.ICC* (siehe (N084.100)) ist, oder
   ii. ICCSN8.int ungleich *iccsn8* (siehe (N019.900)c) ist.
   iii. ICCSN8.int identisch zu ICCSN8.ext ist.
7. Setze KD.i = RAND(64)
8. Setze *R* = RND.int || ICCSN8.int || RND.ext || ICCSN8.ext || KD.i
9. Setze *C2* = 3TDES_CBC_ENC(*affectedObject.encKey*, 0, *R*)
10. Setze *MAC2* = CALCULATE_Retail_MAC(*affectedObject.macKey*, *C2*)
11. Setze *authData* = *C2* || *MAC2*
12. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).

[<=]

G2_N084.500 - (N084.500) K_COS
Wenn das Kommando mit dem Trailer AuthenticationFailure terminiert, dann MUSS clearSecurityStatusKey(*affectedObject*) ausgeführt werden.**[<=]**

G2_N084.600 - (N084.600) K_COS
Wenn das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus(*affectedObject*) ausgeführt werden.**[<=]**

G2_N084.700 - (N084.700) K_COS
Als Datenfeld der Antwortnachricht MUSS der (möglicherweise leere) Oktettstring *authData* verwendet werden.**[<=]**

G2_N084.800 - (N084.800) K_COS
Wenn nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.**[<=]**

A_16507 - (N084.900)a K_TST
Die Priorität der Trailer in CosT_e88 MUSS herstellerspezifisch sein.**[<=]**

G2_N084.900.b - (N084.900)b K_COS
Jeder Trailer in CosT_e88 MUSS eine höhere Priorität als NoError haben.**[<=]**

G2_N084.910 - (N084.910) K_COS
Das COS MUSS spätestens nach dem Versenden der Antwort-APDU und vor dem Empfang der nächsten Kommando-APDU *RND.ICC* auf den Wert "NoRandom" setzen (siehe (N029.900)b).**[<=]**

Hinweis (110): Im Rahmen dieser Spezifikation sind Authentisierungssequenzen nicht unterbrechbar (siehe (N104.600)). Das Authentisierungsprotokoll rsaSessionkey4SM gemäß 15.4.3 führt zunächst ein interne Authentisierung durch, wobei eine Zugriffsregelprüfung für den privaten Schlüssel stattfindet (siehe (N086.800)). Deshalb ergäbe eine Zugriffsregelprüfung gemäß (N084.400)c.2 in allen praxisrelevanten Fällen stets True. Funktional ist eine Zugriffsregelprüfung gemäß (N084.400)c.2 deshalb nicht erforderlich. Möglicherweise ist aber im

~~*Rahmen einer Evaluierung nachzuweisen, dass ein Unterlassen der Zugriffsregelprüfung gemäß (N084.400)c.2 die Sicherheit nicht unzulässig mindert.*~~

~~*Hinweis (111): Bei der Schlüsselsuche in (N084.400)c.1.ii wird nach einem privaten Schlüsselobjekt gesucht. Trotzdem wird* algID *aus* keyReferenceList.externalAuthenticate *entnommen, damit sicher gestellt ist, dass auch das private Schlüsselobjekt den hier zum Tragen kommenden Algorithmus unterstützt. Aus der übrigen Spezifikation ergibt sich, dass dieser Algorithmus Element von {rsaSessionkey4SM, rsaSessionkey4TC} ist.*~~

14.7.2 GENERAL AUTHENTICATE

Das Kommando GENERAL AUTHENTICATE wird in komplexen Authentisierungsprotokollen eingesetzt um die Authentizität einer Smartcard nachzuweisen oder um die Authentizität der "externen Welt" zu prüfen. Allen hier behandelten Authentisierungsprotokollen ist gemeinsam, dass sie aus mehr als einem Schritt bestehen und für jeden Schritt ein GENERAL AUTHENTICATE verwendet wird. Alle Kommandos der hier behandelten Authentisierungsprotokolle sind mittels Command Chaining (siehe 11.8) miteinander verknüpft. Anhand den Einträgen in *channelContext.keyReferenceList* und der Schrittnummer erkennt das COS, welche Aktion ansteht, wie die Daten in der Kommandonachricht zu verarbeiten sind und welche Antwortdaten zu erstellen sind. Von der externen Welt wird gefordert, dass sie die Abfolge des Authentisierungsprotokolls einhält.

14.7.2.1 Gegenseitiges Authentisieren mittels PACE für Endnutzerkarten

Dieser Abschnitt behandelt die Kommandonachrichten für die in CosA_8da mit "CosA_*PICC*" bezeichnete Seite, die in[BSI-TR-03110-2#3.2] "eIDAS token (ICC)" genannt wird. Die Kommandonachrichten für die andere Seite werden in14.7.2.4 beschrieben. Nach erfolgreichem Abschluss dieses Authentisierungsprotokolls liegen in "CosA_*PICC*" Sessionkeys vor, die im Rahmen von Secure Messaging verwendbar sind.

14.7.2.1.1 Use Case PACE für Endnutzerkarten, Schritt 1a

In dieser Variante wird der erste Schritt des PACE Authentisierungsprotokolls für eine Endnutzerkarte durchgeführt. Der komplette Ablauf ist in 15.4.2 beschrieben. Der erste Schritt entspricht [BSI-TR-03110-3#B.1, Step 1, B.1.1] encrypted nonce.

A_16520 - (N085.000) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

a. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
b. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16521 - (N085.001) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_a9a verwendet werden.

Tabelle 233: CosT_a9a: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 1a

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 00'
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.1.2 Use Case PACE für Endnutzerkarten, Schritt 2a

In dieser Variante wird der zweite Schritt von PACE für eine Endnutzerkarte durchgeführt, der [BSI-TR-03110-3#B.1, Step 2, B.1.2.1] entspricht, map nonce.

A_16522 - (N085.002) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

c. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

d. Der Parameter $\sim PK1_{PCD}$ ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.064)b.2 kein Fehler auftritt.

e. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16523 - (N085.003) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_19a verwendet werden.

Tabelle 234: CosT_19a: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 2a

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – L_{7C} – (81 – L_{81} – P2OS($\sim PK1_{PCD}$))', DER codiertes Datenfeld
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.1.3 Use Case PACE für Endnutzerkarten, Schritt 3a

In dieser Variante wird der dritte Schritt von PACE für eine Endnutzerkarte durchgeführt, der [BSI-TR-03110-3#B.1, Step 3] entspricht, perform key agreement.

A_16524 - (N085.004) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

f. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

g. Der Parameter $\sim PK2_{PCD}$ ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.064)c.1 kein Fehler auftritt.

h. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16525 - (N085.005) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_29a verwendet werden.

Tabelle 235: CosT_29a: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 3a

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – L_{7C} – (83 – L_{83} – P2OS($\sim PK2_{PCD}$))', DER codiertes Datenfeld
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.1.4 Use Case PACE für Endnutzerkarten, Schritt 4a

In dieser Variante wird der vierte und letzte Schritt von PACE für eine Endnutzerkarte durchgeführt, der [BSI-TR-03110-3#B.1, Step 4] entspricht, mutual authentication.

A_16526 - (N085.006) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

i. Der Parameter T_{PCD} ist ein Oktettstring der Länge acht mit beliebigem Inhalt.

j. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16527 - (N085.007) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_152 verwendet werden.

Tabelle 236: CosT_152: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 4a

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 0A – (85 – 08 – T_{PCD})'
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.2 Gegenseitiges Authentisieren mittels ELC Schlüsseln

14.7.2.2.1 Use Case gegenseitige ELC-Authentisierung, Schritt 1

In dieser Variante wird der erste Schritt einer gegenseitigen Authentisierung mittels ELC-Schlüssel durchgeführt, bei der auch Sessionkeys ausgehandelt werden. Der komplette Ablauf ist in 15.4.4 beschrieben.

A_16528 - (N085.010) K_externeWelt {K_Karte}
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

k. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

l. Der Parameter *keyRef* MUSS zwölf Oktett lang sein. Er enthält eine Schlüsselreferenz für einen öffentlichen Authentisierungsschlüssel, der mittels CV-Zertifikat importiert wurde.

m. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16529 - (N085.012) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_d3c verwendet werden.

Tabelle 237: CosT_d3c: GENERAL AUTHENTICATE gegenseitige ELC-Authentisierung, Schritt 1

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 0E – (C3 – 0C – *keyRef*)'
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.2.2 Use Case gegenseitige ELC-Authentisierung, Schritt 2

In dieser Variante wird der zweite und letzte Schritt einer gegenseitigen Authentisierung mittels ELC-Schlüssel durchgeführt, bei der auch Sessionkeys ausgehandelt werden.

A_16530 - (N085.014) K_externeWelt {K_Karte}
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:

n. Der Parameter *ephemeralPK_oponent* MUSS einen Punkt auf derselben elliptischen Kurve enthalten, wie der öffentliche Schlüssel, der in Schritt 1 selektiert wurde.

[<=]

A_16531 - (N085.016) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_aca verwendet werden.

Tabelle 238: CosT_aca: GENERAL AUTHENTICATE gegenseitige ELC-Authentisierung, Schritt 2

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – L_{7C} – (85 – L_{85} – *ephemeralPK_oponent*)', DER codiertes Datenfeld

[<=]

14.7.2.3 Authentisieren für asynchrone, symmetrische Kartenadministration

14.7.2.3.1 Use Case Authentisieren für asynchrone, sym. Administration, Schritt 1

In dieser Variante wird der erste Schritt durchgeführt, bei der Sessionkeys symmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben.

A_16532 - (N085.020) K_externeWelt {K_Karte}
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

o. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

p. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16533 - (N085.022) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA__e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_d01 verwendet werden.

Tabelle 239: CosT_d01: GENERAL AUTHENTICATE, asynchrone, symmetrische Administration, Schritt 1

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 02 – (81 – 00)'
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.3.2 Use Case Authentisieren für asynchrone, sym. Administration, Schritt 2

In dieser Variante wird der zweite und letzte Schritt durchgeführt, bei der Sessionkeys symmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben.

A_16534 - (N085.024) K_externeWelt {K_Karte}
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:
a. Der Parameter *cmdData* MUSS 76 Oktett lang sein.**[<=]**

A_16535 - (N085.026) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_0d9 verwendet werden.

Tabelle 240: CosT_0d9: GENERAL AUTHENTICATE, asynchrone, symmetrische Administration, Schritt 2

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 4E – (82 – 4C – *cmdData*)'

[<=]

14.7.2.4 Gegenseitiges Authentisieren mittels PACE für Sicherheitsmodule

Dieser Abschnitt behandelt die Kommandonachrichten für die in CosA_8da mit "COSb PCD" bezeichnete Seite, die in [BSI-TR-03110-2#3.2] "Terminal (PCD)" genannt wird. Die Kommandonachrichten für die andere Seite werden in14.7.2.1 beschrieben. Nach erfolgreichem Abschluss dieses Authentisierungsprotokolls liegen in "COSb PCD" Sessionkeys vor, die im Rahmen von PSO-Kommandos einen Trusted Channel unterstützen.

14.7.2.4.1 Use Case PACE für Sicherheitsmodule, Schritt 1b

In dieser Variante wird der erste Schritt des PACE Authentisierungsprotokolls für ein Sicherheitsmodul durchgeführt. Der komplette Ablauf ist in 15.4.2 beschrieben. Die Karte wird veranlasst ein ephemeres Schlüsselpaar zu generieren.

A_16536 - (N085.030) K_externeWelt {K_Karte}, Option_PACE_PCD
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

q. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

r. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16537 - (N085.031) K_externeWelt {K_Karte}, Option_PACE_PCD
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_441 verwendet werden.

Tabelle 241: CosT_441: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 1b

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 00'
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.4.2 Use Case PACE für Sicherheitsmodule, Schritt 2b

In dieser Variante wird der zweite Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte wird veranlasst die *nonce* der Gegenseite zu rekonstruieren.

A_16538 - (N085.032) K_externeWelt {K_Karte}, Option_PACE_PCD
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

s. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

t. Der Parameter z ist ein Oktettstring der Länge 16 mit beliebigem Inhalt.

u. Der Parameter *CAN* ist ein Oktettstring mit einer Länge aus dem Intervall [1, 16] und beliebigem Inhalt.

[<=]

A_16539 - (N085.033) K_externeWelt {K_Karte}, Option_PACE_PCD
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_b7a verwendet werden.

Tabelle 242: CosT_b7a: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 2b

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – L_{7C} – [(80 – 10 – z) || (C0 – L_{C0} – *CAN*)]', DER codiertes Datenfeld

[<=]

14.7.2.4.3 Use Case PACE für Sicherheitsmodule, Schritt 3b

In dieser Variante wird der dritte Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte wird veranlasst ein ephemeres Schlüsselpaar für ephemere Domainparameter zu generieren.

A_16540 - (N085.034) K_externeWelt {K_Karte}, Option_PACE_PCD
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

v. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

w. Der Parameter $\sim PK1_{PICC}$ ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.066)c.1 kein Fehler auftritt.

x. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16541 - (N085.035) K_externeWelt {K_Karte}, Option_PACE_PCD
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_8b7 verwendet werden.

Tabelle 243: CosT_8b7: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 3b

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – L_{7C} – (82 – L_{82} – P2OS($\sim PK1_{PICC}$))', DER codiertes Datenfeld
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.4.4 Use Case PACE für Sicherheitsmodule, Schritt 4b

In dieser Variante wird der vierte Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte wird veranlasst Sessionkeys abzuleiten.

A_16542 - (N085.036) K_externeWelt {K_Karte}, Option_PACE_PCD
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

y. Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.

z. Der Parameter $\sim PK2_{PICC}$ ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.066)d.1 kein Fehler auftritt.

aa. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

[<=]

A_16543 - (N085.037) K_externeWelt {K_Karte}, Option_PACE_PCD
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_d4f verwendet werden.

Tabelle 244: CosT_d4f: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 4b

	Inhalt	Beschreibung
CLA	'10'	CLA-Byte gemäß [ISO/IEC 7816-4], Chaining Bit b5 gesetzt
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – L_{7C} – (84 – L_{84} – P2OS($\sim PK2_{PICC}$))', DER codiertes Datenfeld
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.2.4.5 Use Case PACE für Sicherheitsmodule, Schritt 5b

In dieser Variante wird der fünfte und letzte Schritt von PACE für ein Sicherheitsmodul durchgeführt. Die Karte überprüft den MAC der Gegenseite.

A_16544 - (N085.038) K_externeWelt {K_Karte}, Option_PACE_PCD
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:
a. Der Parameter T_{PICC} ist ein Oktettstring der Länge acht mit beliebigem Inhalt.**[<=]**

A_16545 - (N085.039) K_externeWelt {K_Karte}, Option_PACE_PCD
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_8fe verwendet werden.

Tabelle 245: CosT_8fe: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 5b

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 0A – (86 – 08 –– T_{PICC})'

[<=]

14.7.2.5 Authentisieren für asynchrone, asym. Kartenadministration

14.7.2.5.1 Use Case Authentisieren für asynchrone, asym. Administration, Schritt 1

In dieser Variante wird der erste Schritt durchgeführt, bei der Sessionkeys asymmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben. Die Kommando-APDU, deren Parameter und deren Bedeutung sind identisch zum Use Case in 14.7.2.2.1, damit gelten hier die Anforderungen (N085.010) und (N085.012).

14.7.2.5.2 Use Case Authentisieren für asynchrone, asym. Administration, Schritt 2

In dieser Variante wird der zweite und letzte Schritt durchgeführt, bei der Sessionkeys asymmetrisch übertragen werden. Der komplette Ablauf ist in 15.5 beschrieben.

A_16546 - (N085.040) K_externeWelt {K_Karte}
Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:

bb. Der Parameter *cmdData* MUSS gemäß (N085.068)b.7 gewählt werden, wobei die Länge von KD.e in (N085.068)b.7.viii 64 Oktett betragen MUSS.

[<=]

A_16547 - (N085.041) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_45a verwendet werden.

Tabelle 246: CosT_45a: GENERAL AUTHENTICATE, asynchrone, asymmetrische Administration, Schritt 2

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'86'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	no information given
P2	'00'	no information given
Data	'XX…XX'	'7C – 81F8 – (82 – 81F5 – *cmdData*)' falls brainpoolP256r1 '7C – 820139 – (82 – 820135 – *cmdData*)' falls brainpoolP384r1 '7C – ~~82013B~~ 82017B – (82 – 820177 – *cmdData*)' falls brainpoolP512r1

[<=]

14.7.2.6 Antwort der Karte auf Generelles Authentisieren

Tabelle 247: CosT_2e0: GENERAL AUTHENTICATE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*responseData*	Antwortdaten vorhanden, möglicherweise leer
Trailer	**Inhalt**	**Beschreibung**
'63 00'	AuthenticationFailure	Authentisierung fehlgeschlagen
'90 00'	NoError	Erfolgreiche Authentisierung

Tabelle 248: CosT_b4e: GENERAL AUTHENTICATE Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	ParameterMismatch	Domainparameter passen nicht zusammen
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	KeyExpired	Gültigkeitszeitraum des Schlüssels ist abgelaufen
'69 85'	NoKeyReference	Keinen symmetrischen Schlüssel ausgewählt
'69 85'	NoPrkReference	Keinen privaten Schlüssel ausgewählt
'6A 80'	NumberPreconditionWrong	Vorbedingung zum Laden des Scenarios nicht erfüllt
'6A 80'	NumberScenarioWrong	Scenario wurde bereits geladen
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 88'	PrKNotFound	privaten Schlüssel nicht gefunden
'6A 88'	KeyNotFound	Authentisierungsschlüssel nicht gefunden

Hinweis CosH_c81: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16548 - (N085.048) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das GENERAL AUTHENTICATE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.7.2.7 Kommandoabarbeitung innerhalb der Karte

G2_N085.050.a - (N085.050)a K_COS
Das COS MUSS die GENERAL AUTHENTICATE-Varianten aus
- 14.7.2.2- Gegenseitiges Authentisieren mittels ELC Schlüsseln,
- 14.7.2.3- Authentisieren für asynchrone, symmetrische Kartenadministration und
- 14.7.2.5- Authentisieren für asynchrone, asym. Kartenadministration

unterstützen.**[<=]**

G2_N085.050.b - (N085.050)b K_COS, Option_kontaktlose_Schnittstelle
Das COS MUSS die GENERAL AUTHENTICATE-Variante aus
- 14.7.2.1- Gegenseitiges Authentisieren mittels PACE für Endnutzerkarten

unterstützen.**[<=]**

G2_N085.050.c - (N085.050)c K_COS, Option_PACE_PCD
Das COS MUSS die GENERAL AUTHENTICATE-Variante aus
- 14.7.2.4- Gegenseitiges Authentisieren mittels PACE für Sicherheitsmodule

unterstützen.**[<=]**

A_16549 - (N085.050)d K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere GENERAL AUTHENTICATE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16550 - (N085.051) K_externeWelt {K_Karte}
Schlüsselauswahl in *channelContext.keyReferenceList*: Die externe Welt MUSS sicherstellen, dass bei Aufruf des Kommandos GENERAL AUTHENTICATE einer der folgenden Fälle vorliegt:
a. (Fall gegenseitige ELC-Authentisierung, Schlüsselauswahl gemäß (N100.900))
   1. In *channelContext.keyReferenceList.internalAuthenticate* ist
      i. eine Schlüsselreferenz eingetragen, die auf ein privates ELC-Schlüsselobjekt verweist und
      ii. dort ist *algID* aus der Menge {elcSessionkey4SM, elcSessionkey4TC} und
   2. *channelContext.keyReferenceList.externalAuthenticate* ist leer.

b. (Fall asynchrone, symmetrische Administration, Schlüsselauswahl gemäß (N102.400))
   1. In *channelContext.keyReferenceList.externalAuthenticate* ist eine Schlüsselreferenz eingetragen und dort ist *algID* gleich aesSessionkey4SM und
   2. *channelContext.keyReferenceList.internalAuthenticate* ist leer.

c. (Fall PACE, Schlüsselauswahl gemäß (N102.448))
   Sowohl in *channelContext.keyReferenceList.internalAuthenticate*, als auch in *channelContext.keyReferenceList.externalAuthenticate*
   1. identische Schlüsselreferenzen und
   2. identische *algID* aus den in (N102.440) genannten Mengen eingetragen.

d. (Fall asynchrone, asymmetrische Administration, Schlüsselauswahl gemäß (N100.900))
   1. In *channelContext.keyReferenceList.internalAuthenticate* ist
      i. eine Schlüsselreferenz eingetragen, die auf ein privates ELC-Schlüsselobjekt verweist und
      ii. dort ist *algID* aus der Menge {elcAsynchronAdmin} und
   2. *channelContext.keyReferenceList.externalAuthenticate* ist leer.**[<=]**

Hinweis CosH_5ba: Gemäß (N085.051) findet das COS in channelContext.keyReferenceList *hinreichend Information anhand derer die diversen Authentisierungsprotokolle unterscheidbar sind. Deshalb ist eine Unterscheidung der hier behandelten Authentisierungsprotokolle anhand von weiteren Kommandoparametern (etwa P1 und/oder P2) nicht erforderlich.*

A_16551 - (N085.052) K_COS
Falls dies der erste Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N085.052.a - (N085.052)a K_COS
Schritt a: Wenn die acht LSByte von *keyRef* identisch sind zu *iccsn8* (siehe (N019.900)c), dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.**[<=]**

A_16552 - (N085.052)b K_COS
Schritt b: Wenn *channelContext.keyReferenceList.internalAuthenticate* nicht leer ist, dann wird *affectedObject_PrK* = SearchKey(
   channelContext. currentFolder,
   channelContext. keyReferenceList.internalAuthenticate.keyReference,
   channelContext. keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N085.052.b.1 - (N085.052)b.1 K_COS
Schritt b.1: Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.**[<=]**

G2_N085.052.b.2 - (N085.052)b.2 K_COS
Schritt b.2: Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

A_16553 - (N085.052)c K_COS
Schritt c: Es wird
affectedObject_PuK = SearchKey(
channelContext. currentFolder,
keyRef,
channelContext. keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N085.052.c.1 - (N085.052)c.1 K_COS
Schritt c.1: Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N085.052.c.2 - (N085.052)c.2 K_COS
Schritt c.2: Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N085.052.d - (N085.052)d K_COS
Schritt d: Wenn *affectedObject_PuK. expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.**[<=]**

G2_N085.052.e - (N085.052)e K_COS
Schritt e: Wenn *affectedObject_PrK. domainParameter* verschieden ist von *affectedObject_PuK. domainParameter*, dann MUSS das Kommando mit dem Trailer ParameterMismatch terminieren.**[<=]**

G2_N085.052.f - (N085.052)f K_COS
Schritt f: Zu den Domainparametern von *affectedObject_PrK* bzw. *affectedObject_PuK* wird ein ephemeres Schlüsselpaar (*ephemeralSK_self*, *ephemeralPK_self*) erzeugt.**[<=]**

G2_N085.052.g - (N085.052)g K_COS
Schritt g: Der private Schlüssel *ephemeralSK_self* MUSS mindestens bis zum nächsten Schritt dieses Authentisierungsprotokolls gespeichert werden.**[<=]**

G2_N085.052.h - (N085.052)h K_COS
Schritt h: Der öffentliche Schlüssel *ephemeralPK_self* MUSS DER codiert wie folgt in die Antwortdaten eingestellt werden:
responseData = '7C-L_{7C}-[85-L_{85}-P2OS(*ephemeralPK_self*, *affectedObject_PrK.domainParameter.L*)]'.**[<=]**

A_16554 - (N085.054) K_COS
Wenn dies der zweite Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist und *algId* == elcSessionkey4SM ist, dann MÜSSEN die in (N085.052) ausgewählten Objekte *affectedObject_PrK* und *affectedObject_PuK* in den folgende Schritten verwendet werden:**[<=]**

G2_N085.054.a - (N085.054)a K_COS

Schritt a: Der ephemere öffentliche Schlüssel *PK_oponent* des Protokollpartners wird wie folgt aus den Kommandodaten extrahiert:
PK_oponent = OS2P(*ephemeralPK_oponent*, *affectedObject_PrK.domainParameter*).**[<=]**

Hinweis CosH_f69: Falls (N085.014)a eingehalten wird, ist diese Operation stets fehlerfrei.

G2_N085.054.b - (N085.054)b K_COS
Schritt b: Wenn AccessRuleEvaluation(*affectedObject_PrK*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N085.054.c - (N085.054)c K_COS
Schritt c: Das COS MUSS die Werte zur Ableitung der Sessionkeys wie folgt berechnen:
1. K1 = ECKAvalue(*affectedObject_PrK.d*, *PK_oponent*, *affectedObject_PrK.domainParameter*).
2. K2 = ECKAvalue(*ephemeralSK_self*, *affectedObject_PuK*, *affectedObject_PrK.domainParameter*).
3. KD.i = K1 || K2
4. KD.e = I2OS(0, OctetLength(KD.i)) = '00…00'.
5. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).**[<=]**

G2_N085.054.d - (N085.054)d K_COS
Schritt d: Es MUSS setSecurityStatus(*affectedObject_PuK*) ausgeführt werden.**[<=]**

G2_N085.054.e - (N085.054)e K_COS
Schritt e: Die Antwortdaten MÜSSEN leer sein: *responseData* = ''.**[<=]**

A_16555 - (N085.056) K_COS, Option_Kryptobox
Wenn dies der zweite Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist und *algId* == elcSessionkey4TC ist, dann MÜSSEN die in (N085.052) ausgewählten Objekte *affectedObject_PrK* und *affectedObject_PuK* in den folgende Schritten verwendet:**[<=]**

G2_N085.056.a - (N085.056)a K_COS, Option_Kryptobox
Schritt a: Der ephemere öffentliche Schlüssel *PK_oponent* des Protokollpartners wird wie folgt aus den Kommandodaten extrahiert:
PK_oponent = *OS2P(ephemeralPK_oponent, affectedObject_PrK.domainParameter)*.**[<=]**

Hinweis CosH_e18: Falls (N085.014)a eingehalten wird, ist diese Operation stets fehlerfrei.

G2_N085.056.b - (N085.056)b K_COS, Option_Kryptobox
Schritt b: Wenn AccessRuleEvaluation(*affectedObject_PrK*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_16556 - (N085.056)c K_COS, Option_Kryptobox
Schritt c: Das COS MUSS die Werte zur Ableitung der Sessionkeys wie folgt berechnen:
1. K1 = ECKAvalue(*ephemeralSK_self*, *affectedObject_PuK*, *affectedObject_PrK.domainParameter*).
2. K2 = ECKAvalue(*affectedObject_PrK.d*, *PK_oponent*, *affectedObject_PrK.domainParameter*).
3. KD.i = K1 || K2
4. KD.e = I2OS(0, OctetLength(KD.i)) = '00…00'.

5. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).**[<=]**

G2_N085.056.d - (N085.056)d K_COS, Option_Kryptobox
Schritt d: Wenn das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus(*affectedObject_PuK*) ausgeführt werden.**[<=]**

G2_N085.056.e - (N085.056)e K_COS, Option_Kryptobox
Schritt e: Die Antwortdaten MÜSSEN leer sein: *responseData* = ''.**[<=]**

A_16557 - (N085.060) K_COS
Wenn dies der erste Schritt des Authentisierungsprotokolls für eine asynchrone, symmetrische Kartenadministration ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N085.060.a.1 - (N085.060)a.1 K_COS
Schritt a.1: Wenn *channelContext.keyReferenceList.externalAuthenticate* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16558 - (N085.060)a.2 K_COS
Schritt a.2: Wenn *channelContext.keyReferenceList.externalAuthenticate* nicht leer ist, dann wird
affectedObject = SearchKey(
channelContext. currentFolder,
keyReferenceList externalAuthenticate.keyReference,
keyReferenceList.externalAuthenticate.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N085.060.a.2.i - (N085.060)a.2.i K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N085.060.a.2.ii - (N085.060)a.2.ii K_COS
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N085.060.b - (N085.060)b K_COS
Schritt b: Es gilt: *responseData* = '7C – 04 – (81 – 02 – I2OS(*affectedObject. numberScenario*))'.**[<=]**

A_16559 - (N085.062) K_COS
Wenn dies der zweite Schritt des Authentisierungsprotokolls für eine asynchrone, symmetrische Kartenadministration ist, dann MUSS das in (N085.060) ausgewählten Objekte *affectedObject* in den folgenden Schritten verwendet werden:**[<=]**

G2_N085.062.a - (N085.062)a K_COS
Schritt a: Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_16560 - (N085.062)b K_COS
Schritt b: *cmdData* MUSS wie folgt aufgeteilt werden:
1. *cmdData* == *M* || *MAC*.
2. OctetLength(*MAC*) == 8.**[<=]**

A_16561 - (N085.062)c K_COS

Schritt c: *out* = VerifyCMAC_IsoPadding(*affectedObject.macKey*, *MAC*, *M*) **[<=]**

G2_N085.062.d - (N085.062)d K_COS
Schritt d: Falls *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.**[<=]**

A_16562 - (N085.062)e K_COS
Schritt e: Die Nachricht *M* MUSS wie folgt aufgeteilt werden:
1. *M* == *NumberPrecondition* || *NumberScenario* || *C*.
2. OctetLength(*NumberPrecondition*) == 2.
3. OctetLength(*NumberScenario*) == 2.
4. OctetLength(*C*) == 64.**[<=]**

G2_N085.062.f.1 - (N085.062)f.1 K_COS
Schritt f.1: Wenn *affectedObject*. *numberScenario* kleiner als OS2I(*NumberPrecondition*) ist, genau dann MUSS das Kommando mit dem Trailer NumberPreconditionWrong terminieren.**[<=]**

G2_N085.062.f.2 - (N085.062)f.2 K_COS
Schritt f.2: Wenn *affectedObject*. *numberScenario* größer gleich OS2I(*NumberScenario*) ist, genau dann MUSS das Kommando mit dem Trailer NumberScenarioWrong terminieren.**[<=]**

G2_N085.062.g - (N085.062)g K_COS
Schritt g: *affectedObject*. *numberScenario* MUSS transaktionsgesichert auf den Wert OS2I(*NumberScenario*) gesetzt werden.**[<=]**

G2_N085.062.h - (N085.062)h K_COS
Schritt h: Es gilt:
1. KD.e = AES_CBC_DEC(*affectedObject.encKey*, 0, *C*).
2. KD.i = '00…00' = I2OS(0, OctetLength(KD.e)).
3. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).**[<=]**

G2_N085.062.i - (N085.062)i K_COS
Schritt i: Wenn das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus(*affectedObject*) ausgeführt werden.**[<=]**

G2_N085.062.j - (N085.062)j K_COS
Schritt j: Die Antwortdaten MÜSSEN leer sein: *responseData* = ''.**[<=]**

A_16563 - (N085.064) K_COS, Option_kontaktlose_Schnittstelle
PACE Authentisierungsprotokoll für eine Endnutzerkarte: Falls in *channelContext.keyReferenceList.internalAuthenticate.algID* ein Algorithmus aus der in (N102.440)a genannten Menge eingetragen ist, dann MUSS folgendes ausgeführt werden:**[<=]**

A_16564 - (N085.064)a K_COS, Option_kontaktlose_Schnittstelle
Falls dies der erste Schritt des Authentisierungsprotokolls, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

A_16565 - (N085.064)a.1 K_COS, Option_kontaktlose_Schnittstelle
Schlüsselsuche, es wird
affectedObject = SearchKey(
channelContext. currentFolder,
channelContext. keyReferenceList.internalAuthenticate.keyReference,
channelContext. keyReferenceList.internalAuthenticate.algID

) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N085.064.a.1.i - (N085.064)a.1.i K_COS, Option_kontaktlose_Schnittstelle
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N085.064.a.1.ii - (N085.064)a.1.ii K_COS, Option_kontaktlose_Schnittstelle
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N085.064.a.2-6 - (N085.064)a.2 K_COS, Option_kontaktlose_Schnittstelle
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_16566 - (N085.064)a.3 K_COS, Option_kontaktlose_Schnittstelle
Das Attribut *affectedObject.can* (siehe (N017.028)) wird in einen Oktettstring gewandelt (vergleiche [BSI-TR-03110-3#D.2.1.4]): Die Ziffernfolge *can* besteht aus den Ziffern z_1, …, z_n und MUSS in einen Oktettstring *CAN* = 'o_1, …, o_n' umgewandelt werden, indem jede Zifferz_i in ein Oktett o_i transformiert wird mit
i. z_i = 0 =>o_i = '30',
ii. z_i = 1 =>o_i = '31',
iii. z_i = 2 =>o_i = '32',
…
ix. z_i = 9 =>o_i = '39',**[<=]**

A_16567 - (N085.064)a.4-6 K_COS, Option_kontaktlose_Schnittstelle
4. *s* = RAND(16),
5. *z* = AES_ENC(KDF(*CAN*, 3, *channelContext.keyReferenceList.internalAuthenticate.algID*), *s*),
6. *responseData* = '7C – 12 – (80 – 10 –*z*)'.**[<=]**

A_16568 - (N085.064)b K_COS, Option_kontaktlose_Schnittstelle
Falls dies der zweite Schritt des Authentisierungsprotokolls ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

A_16569 - (N085.064)b.1 K_COS, Option_kontaktlose_Schnittstelle
Es MUSS eine Zuordnung von *channelContext. keyReferenceList.internalAuthenticate.algID* zu *dP* durchgeführt werden (diese Zuordnung gilt auch für die übrigen Schritte des Protokolls):**[<=]**

G2_N085.064.b.1.i - (N085.064)b.1.i K_COS, Option_kontaktlose_Schnittstelle
id-PACE-ECDH-GM-AES-CBC-CMAC-128 => *dP* = brainpoolP256r1**[<=]**

G2_N085.064.b.1.ii - (N085.064)b.1.ii K_COS, Option_kontaktlose_Schnittstelle
id-PACE-ECDH-GM-AES-CBC-CMAC-192 => *dP* = brainpoolP384r1**[<=]**

G2_N085.064.b.1.iii - (N085.064)b.1.iii K_COS, Option_kontaktlose_Schnittstelle
id-PACE-ECDH-GM-AES-CBC-CMAC-256 => *dP* = brainpoolP512r1**[<=]**

G2_N085.064.b.2 - (N085.064)b.2 K_COS, Option_kontaktlose_Schnittstelle
Aus dem Wertfeld $value_{81}$ des DO81 im Kommandodatenfeld wird ein Punkt extrahiert: $\sim PK1_{PCD}$ = OS2P($value_{81}$, *dP*), **[<=]**

G2_N085.064.b.3-6 - (N085.064)b.3-6 K_COS, Option_kontaktlose_Schnittstelle

b.3: Es wird ein ephemeres Schlüsselpaar generiert:
i. $\sim SK1_{PICC}$ = zufällige Zahl aus dem Intervall [1, $dP.\,n$ – 1],
ii. $\sim PK1_{PICC}$ = $\sim SK1_{PICC}$ * $dP.\,G$,
b.4: Berechne den ephemeren Basispunkt $\sim G$ mit s aus Schritt 1:
$\sim G$ = s * $dP.\,G$ + ECKApoint($\sim SK1_{PICC}$, $\sim PK1_{PCD}$, dP) ~~$\sim SK1_{PICC}$ * $\sim PK1_{PCD}$~~ ,
b.5: Berechne ephemere Domainparameter $\sim D$ aus dP und $\sim G$:
$\sim D$ = ($dP.\,p$, $dP.\,a$, $dP.\,b$, $\sim G$, $dP.\,n$, $dP.\,h$, $dP.\,L$, $dP.\,t$, $dP.\,OID$),
b.6: *responseData* MUSS ein DER codiertes DO7C wie folgt enthalten:
responseData = '7C – L_{7C} – (82 – L_{82} – P2OS($\sim PK1_{PICC}$, $dP.\,t$ / 8))'**[<=]**

G2_N085.064.c - (N085.064)c K_COS, Option_kontaktlose_Schnittstelle
Falls dies der dritte Schritt des Authentisierungsprotokolls ist, dann MÜSSEN folgende Schritte ausgeführt werden:

1. Aus dem Wertfeld $value_{83}$ des DO83 im Kommandodatenfeld wird ein Punkt extrahiert:
   $\sim PK2_{PCD}$ = OS2P($value_{83}$, $\sim D$),
2. Es wird ein weiteres ephemeres Schlüsselpaar generiert:
   i. $\sim SK2_{PICC}$ = zufällige Zahl aus dem Intervall [1, $\sim D.\,n$ – 1],
   ii. $\sim PK2_{PICC}$ = $\sim SK2_{PICC}$ * $\sim D.\,G$,
3. *responseData* MUSS ein DER codieretes DO7C wie folgt enthalten:
   responseData = '7C – L_{7C} – (84 – L_{84} – P2OS($\sim PK2_{PICC}$, $\sim D.\,t$ / 8))'

[<=]

A_16570 - (N085.064)d K_COS, Option_kontaktlose_Schnittstelle
Falls dies der vierte Schritt des Authentisierungsprotokolls ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N085.064.d.1-5 - (N085.064)d.1-5 K_COS, Option_kontaktlose_Schnittstelle

1. Berechne gemeinsames Geheimnis zur Ableitung von Sessionkeys:
   i. KD.i = K_{AB} = ECKAvalue($\sim SK2_{PICC}$, $\sim PK2_{PCD}$, $\sim D$),
   ii. KD.e = I2OS(0, OctetLength(KD.i)) = '00…00',
2. Die *algID* aus *channelContext.keyReferenceList.internalAuthenticate* bestimmt, wie der Schlüssel k_{MAC} berechnet wird: k_{MAC} = KDF(KD.i, 2, *algID*),
3. Für den Zusammenhang zwischen *algID* aus *externalAuthenticate* oder *internalAuthenticate* in *channelContext.keyReferenceList* und *OID* gilt hier *algID* = *OID*.
4. Berechne den DER codierten Oktettstring M_{PCD} gemäß:
   M_{PCD} = '7F49 – L_{7F49} – [(06 – 0A – *OID*) || (86 – L_{86} – P2OS($\sim PK2_{PICC}$, $\sim D$))]',
5. Berechne den DER codierten Oktettstring M_{PICC} gemäß:
   M_{PICC} = '7F49 – L_{7F49} – [(06 – 0A – *OID*) || (86 – L_{86} – P2OS($\sim PK2_{PCD}$, $\sim D$))]',

[<=]

A_16571 - (N085.064)d.6 K_COS, Option_kontaktlose_Schnittstelle
Der MAC aus der Kommandonachricht MUSS überprüft werden:
result = VerifyCMAC_NoPadding(k_{MAC}, T_{PCD}, M_{PCD}) **[<=]**

G2_N085.064.d.6.i - (N085.064)d.6.i K_COS, Option_kontaktlose_Schnittstelle
Wenn *result* den Wert INVALID besitzt, dann MUSS
A. clearSecurityStatusKey(*affectedObject*) ausgeführt werden und
B. das Kommando mit dem Trailer AuthenticationFailure terminieren.**[<=]**

G2_N085.064.d.6.ii - (N085.064)d.6.ii K_COS, Option_kontaktlose_Schnittstelle
Wenn *result* den Wert VALID besitzt, dann
A. MUSS der MAC für die Antwortnachricht wie folgt berechnet werden:
T_{PICC} = CalculateCMAC_NoPadding(k_{MAC}, M_{PICC}), und
B. MUSS setSecurityStatus(*affectedObject*) ausgeführt werden und
C. MÜSSEN die Oktettstrings KD.e und KD.i an den Secure Messaging Layer übergeben werden (siehe CosT_a2e), und
D. MUSS das Datenfeld der Antwortnachricht wie folgt berechnet werden:
responseData = '7C – 0A – (86 – 08 –T_{PICC})'.**[<=]**

A_16572 - (N085.066) K_COS, Option_PACE_PCD
PACE Authentisierungsprotokoll für ein Sicherheitsmodul: Falls in *channelContext.keyReferenceList.internalAuthenticate.algID* ein Algorithmus aus der in (N102.440)b genannten Menge eingetragen ist, dann MUSS folgendes ausgeführt werden:**[<=]**

A_16573 - (N085.066)a K_COS, Option_PACE_PCD
Falls dies der erste Schritt des Authentisierungsprotokolls, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

A_16574 - (N085.066)a.1 K_COS, Option_PACE_PCD
Schlüsselsuche, es wird
affectedObject = SearchKey(
channelContext. currentFolder,
channelContext. keyReferenceList.internalAuthenticate.keyReference,
channelContext. keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N085.066.a.1.i - (N085.066)a.1.i K_COS, Option_PACE_PCD
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N085.066.a.1.ii - (N085.066)a.1.ii K_COS, Option_PACE_PCD
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N085.066.a.2 - (N085.066)a.2 K_COS, Option_PACE_PCD
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N085.066.a.3-5 - (N085.066)a.3-5 K_COS, Option_PACE_PCD
3. Es MUSS eine Zuordnung von *channelContext. keyReferenceList.internalAuthenticate.algID* zu *dP* durchgeführt werden (diese Zuordnung gilt auch für die übrigen Schritte des Protokolls):
   i. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128 => *dP* = brainpoolP256r1
   ii. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192 => *dP* = brainpoolP384r1
   iii. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256 => *dP* = brainpoolP512r1
4. Es wird ein ephemeres Schlüsselpaar generiert:
   i. ~$SK1_{PCD}$ = zufällige Zahl aus dem Intervall [1, *dP. n* – 1],
   ii. ~$PK1_{PCD}$ = ~$SK1_{PCD}$ * *dP. G*,
5. *responseData* MUSS ein DER codieretes DO7C wie folgt enthalten:
responseData = '7C – L_{7C} – (81 – L_{81} – P2OS(~$PK1_{PCD}$, *dP. t* / 8))'**[<=]**

G2_N085.066.b - (N085.066)b K_COS, Option_PACE_PCD

Falls dies der zweite Schritt des Authentisierungsprotokolls ist, dann MÜSSEN folgende Schritte ausgeführt werden:
1. Aus dem Datenfeld der Kommandonachricht werden z und *CAN* extrahiert.
2. s = OS2I(AES_DEC(KDF(*CAN*, 3, *channelContext.keyReferenceList.internalAuthenticate.algID*), z)),
3. *responseData* = '' (leerer Oktettstring).**[<=]**

A_16575 - (N085.066)c K_COS, Option_PACE_PCD
Falls dies der dritte Schritt des Authentisierungsprotokolls ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N085.066.c.1 - (N085.066)c.1 K_COS, Option_PACE_PCD
Aus dem Wertfeld $value_{82}$ des DO82 im Kommandodatenfeld wird ein Punkt extrahiert:
$\sim PK1_{PICC}$ = OS2P($value_{82}$, *dP*), **[<=]**

G2_N085.066.c.2-5 - (N085.066)c.2-5 K_COS, Option_PACE_PCD
c.2: Berechne den ephemeren Basispunkt ~*G* mit s aus Schritt 2:
~*G* = s * *dP*. *G* + ECKApoint($\sim SK1_{PCD}$, $\sim PK1_{PICC}$, *dP*) ~~$\sim SK1_{PCD} * \sim PK1_{PICC}$~~ ,
c.3: Berechne ephemere Domainparameter ~*D* aus *dP* und ~*G*:
~*D* = (*dP*. *p*, *dP*. *a*, *dP*. *b*, ~*G*, *dP*. *n*, *dP*. *h*, *dP*. *L*, *dP*. t, *dP*. *OID*),
c.4: Es wird ein weiteres ephemeres Schlüsselpaar generiert:
i. $\sim SK2_{PCD}$ = zufällige Zahl aus dem Intervall [1, ~*D*. *n* – 1],
ii. $\sim PK2_{PCD} = \sim SK2_{PCD} * \sim D.\ G$,
c.5: *responseData* MUSS ein DER codieretes DO7C wie folgt enthalten:
responseData = '7C – L_{7C} – (83 – L_{83} - P2OS($\sim PK2_{PCD}$, ~*D*. t / 8))'**[<=]**

A_16576 - (N085.066)d K_COS, Option_PACE_PCD
Falls dies der vierte Schritt des Authentisierungsprotokolls, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N085.066.d.1 - (N085.066)d.1 K_COS, Option_PACE_PCD
Aus dem Wertfeld $value_{84}$ des DO84 im Kommandodatenfeld wird ein Punkt extrahiert:
$\sim PK2_{PICC}$ = OS2P($value_{84}$, ~*D*), **[<=]**

G2_N085.066.d.2-7 - (N085.066)d.2-7 K_COS, Option_PACE_PCD
d.2: Berechne gemeinsames Geheimnis zur Ableitung von Sessionkeys:
i. KD.i = K_{AB} = ECKAvalue($\sim SK2_{PCD}$, $\sim PK2_{PICC}$, ~*D*),
ii. KD.e = I2OS(0, OctetLength(KD.i)) = '00…00',
d.3: Die *algID* aus *channelContext.keyReferenceList.internalAuthenticate* bestimmt, wie der Schlüssel k_{MAC} berechnet wird: k_{MAC} = KDF(KD.i, 2, *algID*),
d.4: Für den Zusammenhang zwischen *algID* aus *externalAuthenticate* oder *internalAuthenticate* in *channelContext.keyReferenceList* und *OID* gilt hier *:*
i. *algID* = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128 => *OID* = id-PACE-ECDH-GM-AES-CBC-CMAC-128
ii. *algID* = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192 => *OID* = id-PACE-ECDH-GM-AES-CBC-CMAC-192
iii. *algID* = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256 => *OID* = id-PACE-ECDH-GM-AES-CBC-CMAC-256 ~~128~~
d.5: Berechne den Oktettstring M_{PCD} identisch zu (N085.064)d.4 und den zugehörigen MAC:
T_{PCD} = CalculateCMAC_NoPadding(k_{MAC}, M_{PCD}).
d.6: Berechne den Oktettstring M_{PICC} identisch zu (N085.064)d.5.
d.7: Setze *responseData* = '7C – 0A – (85 – 08 –T_{PCD})'**[<=]**

A_16577 - (N085.066)e K_COS, Option_PACE_PCD
Falls dies der fünfte Schritt des Authentisierungsprotokolls ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

A_16578 - (N085.066)e.1 K_COS, Option_PACE_PCD
Der MAC aus der Kommandonachricht MUSS überprüft werden:
result = VerifyCMAC_NoPadding(k_{MAC}, T_{PICC}, M_{PICC}) **[<=]**

G2_N085.066.e.1.i - (N085.066)e.1.i K_COS, Option_PACE_PCD
Wenn *result* den Wert INVALID besitzt, dann MUSS
A. clearSecurityStatusKey(*affectedObject*) ausgeführt werden und
B. das Kommando mit dem Trailer AuthenticationFailure terminieren.**[<=]**

G2_N085.066.e.1.ii - (N085.066)e.1.ii K_COS, Option_PACE_PCD
Wenn *result* den Wert VALID besitzt, dann
A. MUSS setSecurityStatus(*affectedObject*) ausgeführt werden und
B. MÜSSEN die Oktettstrings KD.e und KD.i an den Secure Messaging Layer übergeben werden (siehe CosT_a2e), und
C. MUSS gelten *responseData* = '' (leerer Oktettstring).**[<=]**

A_16579 - (N085.068)a K_COS
Wenn dies der erste Schritt des Authentisierungsprotokolls für eine asynchrone, asymmetrische Kartenadministration ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N085.068.a.1 - (N085.068)a.1 K_COS
Schlüsselsuche nach einem privaten Schlüsselobjekt, es wird
affectedObject_PrK = SearchKey(
channelContext. currentFolder,
channelContext. keyReferenceList.internalAuthenticate.keyReference,
channelContext. keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler
i. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.
ii. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N085.068.a.2 - (N085.068)a.2 K_COS
Schlüsselsuche nach einem öffentlichen Schlüsselobjekt, es wird
affectedObject_PuK = SearchKey(
channelContext. currentFolder,
keyRef,
channelContext. keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler
i. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
ii. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N085.068.a.3 - (N085.068)a.3 K_COS
Wenn *affectedObject_PuK. expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.**[<=]**

G2_N085.068.a.4 - (N085.068)a.4 K_COS
Wenn *affectedObject_PrK. domainParameter* verschieden ist von *affectedObject_PuK. domainParameter*, dann MUSS das Kommando mit dem Trailer ParameterMismatch terminieren.**[<=]**

G2_N085.068.a.5 - (N085.068)a.5 K_COS
Es gilt: *responseData* = '7C – 04 – (81 – 02 – I2OS(*affectedObject_PrK. numberScenario*))'.**[<=]**

A_16580 - (N085.068)b K_COS
Wenn dies der zweite Schritt des Authentisierungsprotokolls für eine asynchrone, asymmetrische Kartenadministration ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N085.068.b.1 - (N085.068)b.1 K_COS
Wenn AccessRuleEvaluation(*affectedObject_PrK*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_16581 - (N085.068)b.2 K_COS
cmdData MUSS wie folgt aufgeteilt werden:
i. *cmdData* == *M* || *SIG*.
ii. In Abhängigkeit von den *affectedObject_PrK. domainParameter* gilt:
  A. brainpoolP256r1: OctetLength(*SIG*) == 64, *hash* = SHA_256(*M*)
  B. brainpoolP384r1: OctetLength(*SIG*) == 96, *hash* = SHA_384(*M*)
  C. brainpoolP512r1: OctetLength(*SIG*) == 128, *hash* = SHA_512(*M*)
iii. *SIG* == *R* || *S*
iv. OctetLength(*R*) == OctetLength(*S*).**[<=]**

G2_N085.068.b.2-3 - (N085.068)b.3 K_COS
Falls ELC_VER_SIG(*affectedObject_PuK*, *R*, *S*, *hash*) den Wert False besitzt, MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.**[<=]**

A_16582 - (N085.068)b.4 K_COS
Die Nachricht *M* MUSS wie folgt aufgeteilt werden:
i. *M* == *NumberPrecondition* || *NumberScenario* || *cipher*.
ii. OctetLength(*NumberPrecondition*) == 2.
iii. OctetLength(*NumberScenario*) == 2.**[<=]**

G2_N085.068.b.4-5.i - (N085.068)b.5.i K_COS
Wenn *affectedObject_PrK. numberScenario* kleiner als OS2I(*NumberPrecondition*) ist, genau dann MUSS das Kommando mit dem Trailer NumberPreconditionWrong terminieren.**[<=]**

G2_N085.068.b.4-5.ii - (N085.068)b.5.ii K_COS
Wenn *affectedObject_PrK. numberScenario* größer gleich OS2I(*NumberScenario*) ist, genau dann MUSS das Kommando mit dem Trailer NumberScenarioWrong terminieren.**[<=]**

G2_N085.068.b.6 - (N085.068)b.6 K_COS
affectedObject_PrK. numberScenario MUSS transaktionsgesichert auf den Wert OS2I(*NumberScenario*) gesetzt werden.**[<=]**

G2_N085.068.b.7 - (N085.068)b.7 K_COS
Es gilt (*Hinweis CosH_310:* cipher *ist hier identisch zu (N090.300)c, (N091.700)d und (N094.400)c definiert*):
i. *cipher* MUSS ein DER codiertes DOA6 sein.
ii. *cipher* = 'A6 – L_{A6} – (*oidDO* || *keyDO* || *cipherDO* || *macDO*)'.

iii. *oidDO* = '06 – L_{06} – *oid* '.
iv. *keyDO* = '7F49 – L_{7F49} – (86 – L_{86} – PO_A)'.
v. *cipherDO* = '86 – L_{86} – (02 || *C*)'.
vi. *macDO* = '8E – L_{8E} – *T* '.
vii. Falls *oid* verschieden ist zur OID, die gemäß (N008.600)d zu *affectedObject_PrK. privateElcKey. domainParameter* gehört, dann MUSS das Kommando mit dem Trailer ParameterMismatch terminieren. Diese Domainparameter werden im Folgenden mit *dP* bezeichnet.
viii. KD.e = ELC_DEC(PO_A, *affectedObject_PrK.privateElcKey*, *C*, *T*).
ix. KD.i = '00…00' = I2OS(0, OctetLength(KD.e)).
x. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).**[<=]**

G2_N085.068.b.8 - (N085.068)b.8 K_COS
Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS setSecurityStatus(*affectedObject_PuK*) ausgeführt werden.**[<=]**

G2_N085.068.b.9 - (N085.068)b.9 K_COS
Die Antwortdaten MÜSSEN leer sein: *responseData* = ''.**[<=]**

Hinweis CosH_6bc: Wegen (N008.600)d und CosT_a91 ist L_{06} in (N085.068)b.7.iii stets 9 und damit oidDO *stets 11 Oktett lang.*

Hinweis CosH_361: In Abhängigkeit von den Domainparametern von affectedObject_PrK*, (N000.300)a und (N008.600)b gilt für L_{86} in (N085.068)b.7.iv:*
a. brainpoolP256r1: L_{86} = 65 => keyDO ist 70 Oktett lang.
b. brainpoolP384r1: L_{86} = 97 => keyDO ist 102 Oktett lang.
c. brainpoolP512r1: L_{86} = 129 => keyDO ist 136 Oktett lang.

Hinweis CosH_61b: Wegen (N085.040) und (N004.800)e ist L_{86} in (N085.068)b.7.v gleich 1 + 64 + 16 = 81 und damit cipherDO *83 Oktett lang.*

Hinweis CosH_8sd: Wegen (N002.810)h ist L_{8E} in (N085.068)b.7.vi gleich 8 und damit macDO *10 Oktett lang.*

G2_N085.070 - (N085.070) K_COS
Als Datenfeld der Antwortnachricht MUSS der (möglicherweise leere) Oktettstring *responseData* verwendet werden.**[<=]**

G2_N085.072 - (N085.072) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16583 - (N085.074)a K_TST
Die Priorität der Trailer in CosT_b4e MUSS herstellerspezifisch sein.**[<=]**

G2_N085.074.b - (N085.074)b K_COS
Jeder Trailer in CosT_b4e MUSS eine höhere Priorität als NoError haben.**[<=]**

Hinweis CosH_a4e: Im Rahmen dieser Spezifikation sind Authentisierungssequenzen nicht unterbrechbar (siehe (N104.600)).

Hinweis CosH_0cc: Bei der Schlüsselsuche in (N085.052)c wird nach einem öffentlichen Schlüsselobjekt gesucht. Trotzdem wird algID aus keyReferenceList.internalAuthenticate *entnommen, damit sichergestellt ist, dass öffentliches wie privates Schlüsselobjekt denselben Algorithmus verwenden. Aus der übrigen Spezifikation ergibt sich, dass dieser Algorithmus Element der Menge*
{ elcSessionkey4SM, elcSessionkey4TC } ist.

14.7.3 GET SECURITY STATUS KEY

Hinweis CosH_364: Dieses Kommando ist in der Normenreihe ISO/IEC 7816 nicht enthalten. Es ließe sich kombinieren mit dem Kommando EXTERNAL AUTHENTICATE.

Hinweis CosH_149: Bei der Spezifikation der Kommando-APDU wurde das MSE-Set-Kommando zugrunde gelegt.

Hinweis CosH_3ab: Motivation für das Kommando: Ausgehend von der Annahme, dass ein bestimmtes Kommando mit dem Trailer SecurityStatusNotSatisfied terminiert, ist für die steuernde Software gelegentlich nicht offensichtlich, was zu tun ist, um Zugriff zu erhalten. Selbst bei Kenntnis der Zugriffsregel des Objektes lässt sich lediglich beurteilen, ob Secure Messaging Vorgaben eingehalten wurden. Falls eine PIN-Verifikation erforderlich ist, so lässt sich der zugehörige Sicherheitszustand mittels GET PIN STATUS erfragen. Falls aber mehrere Sicherheitszustände von Schlüsseln ins Spiel kommen, dann hat die steuernde Software nur zwei Möglichkeiten: Entweder "Try and error", was schlecht für die Performanz ist, oder sie baut die COS spezifische Zustandsmaschine für den Sicherheitsstatus von Schlüsseln nach (und hofft, dass interner und externer Zustand synchron bleiben).

Das Kommando GET SECURITY STATUS KEY wird verwendet, um den Sicherheitszustand von Schlüsselobjekten zu erfragen. Welcher Sicherheitszustand ausgelesen wird, bestimmt eine Referenz, die als Parameter in den Kommandodaten enthalten ist.

14.7.3.1 Use Case Auslesen Sicherheitsstatus symmetrischer Schlüssels, Option_DES

Die folgenden Punkte sind absichtlich leer: (N085.100), (N085.200), (N085.300), (N085.400).

~~In dieser Variante enthält die APDU des GET SECURITY STATUS KEY Kommandos einen Parameter:~~

A_16508 - (N085.100) K_externeWelt {K_Karte}, Option_DES
~~Der Parameter *cmdData* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.~~**[<=]**

A_16509 - (N085.200) K_externeWelt {K_Karte}, Option_DES
~~Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_0f6 verwendet werden.~~

Tabelle 249, CosT_0f6: GET SECURITY STATUS KEY, symmetrischer Schlüssel

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'82'	Instruction Byte (derselbe Wert wie bei EXTERNAL AUTHENTICATE)
P1	'80'	Auslesen Sicherheitszustand
P2	'00'	–
Data	'XX…XX'	'83 01 || *cmdData* '

[<=]

14.7.3.2 Use Case Auslesen Sicherheitsstatus einer Rolle, Option_RSA_CVC

In dieser Variante enthält die APDU des GET SECURITY STATUS KEY Kommandos einen Parameter:

A_16510 - (N085.300) K_externeWelt {K_Karte}, Option_RSA_CVC
Der Parameter *cmdData* enthält eine Rollenkennung. Wert und Codierung MÜSSEN gemäß 7.1.1.4 und (N005.700) gewählt werden.**[<=]**

A_16511 - (N085.400) K_externeWelt {K_Karte}, Option_RSA_CVC
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_0b6 verwendet werden.

Tabelle 250, CosT_0b6: GET SECURITY STATUS KEY, Rollenkennung

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'82'	Instruction Byte (derselbe Wert wie bei EXTERNAL AUTHENTICATE)
P1	'80'	Auslesen Sicherheitszustand
P2	'00'	–
Data	'XX…XX'	'5F4C - 07 - *cmdData* '

[<=]

14.7.3.3 Use Case Auslesen Sicherheitsstatus einer Bitliste

In dieser Variante enthält die APDU des GET SECURITY STATUS KEY Kommandos zwei Parameter:

A_16512 - (N085.440) K_externeWelt {K_Karte}
Der Parameter *oid* MUSS eine OID aus der Menge {oid_cvc_fl_ti, oid_cvc_fl_cms} enthalten.**[<=]**

A_16513 - (N085.442) K_externeWelt {K_Karte}
Der Parameter *cmdData* MUSS eine sieben Oktett lange Flagliste enthalten. Die beiden höchstwertigen Bit in *cmdData* MÜSSEN den Wert 0 besitzen. Für die Werte übrigen Bit gibt es keine Beschränkung.**[<=]**

A_16514 - (N085.444) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_3cf verwendet werden.

Tabelle 251: CosT_3cf: GET SECURITY STATUS KEY, bitSecurityList

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'82'	Instruction Byte (derselbe Wert wie bei EXTERNAL AUTHENTICATE)
P1	'80'	Auslesen Sicherheitszustand
P2	'00'	–
Data	'XX…XX'	'7F4C – 13 – (06 – 08 – *oid* || 53 – 07 – *cmdData*)'

[<=]

14.7.3.4 Antwort der Karte auf Auslesen Sicherheitsstatus eines Schlüssels

Tabelle 252: CosT_380: GET SECURITY STATUS KEY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 CF'	NoAuthentication	Authentisierungsstatus ist nicht gesetzt
'90 00'	NoError	Authentisierungsstatus ist gesetzt

~~**Tabelle 253, CosT_599: GET SECURITY STATUS KEY Antwort-APDU im Fehlerfall**~~

~~Trailer~~	~~Inhalt~~	~~Beschreibung~~
~~'69 82'~~	~~SecurityStatusNotSatisfied~~	~~Zugriffsregel nicht erfüllt~~
~~'6A 88'~~	~~KeyNotFound~~	~~Adressiertes Schlüsselobjekt wurde nicht gefunden~~

Hinweis CosH_000: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16515 - (N085.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das GET SECURITY STATUS KEY-Kommando zusätzliche Trailer verwendet.**[<=]**

14.7.3.5 Kommandoabarbeitung innerhalb der Karte

G2_N085.600.a - (N085.600)a K_COS
Das COS MUSS die GET SECURITY STATUS KEY-Varianten aus
- ~~14.7.3.1- Use Case Auslesen Sicherheitsstatus symmetrischer Schlüssels, Option_DES~~
- ~~14.7.3.2- Use Case Auslesen Sicherheitsstatus einer Rolle, Option_RSA_CVC~~
- 14.7.3.3- Use Case Auslesen Sicherheitsstatus einer Bitliste

unterstützen.**[<=]**

A_16516 - (N085.600)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere GET SECURITY STATUS KEY-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

(N085.700) ist absichtlich leer.
(N085.800) ist absichtlich leer.

A_16517 - (N085.700) K_COS, Option_DES
~~Wemm *cmdData* die Referenz eines symmetrischen Schlüssels gemäß (N085.200) ist, dann MÜSSEN folgende Schritte ausgeführt werden:~~**[<=]**

G2_N085.700.a - (N085.700)a K_COS
~~Schritt 1: Es wird *affectedObject* = SearchSecretKey(~~
~~*channelContext.currentFolder,*~~
~~*cmdData,*~~
~~Wildcard~~
~~) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.~~
~~*Hinweis CosH_54b: Der Fehler notSupported ist wegen der Wildcard-Suche nicht möglich.*~~**[<=]**

G2_N085.700.b - (N085.700)b K_COS
~~Schritt 2:Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.~~**[<=]**

G2_N085.700.c - (N085.700)c K_COS
~~Schritt 3: Wenn *affectedObject* in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist, dann MUSS als Trailer NoError verwendet werden.~~**[<=]**

G2_N085.800 - (N085.800) K_COS, Option_RSA_CVC
~~Falls *cmdData* eine Rolle gemäß (N085.400) ist und diese Rolle in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist, dann MUSS als Trailer NoError verwendet werden.~~**[<=]**

G2_N085.900 - (N085.900) K_COS
Wenn *cmdData* ein CHAT enthält und es gibt mindestens ein Element in *bitSecurityList* (siehe (N029.900)h), in welchem dieselbe OID und mindestens dieselben Bits gesetzt sind wie in *flagList* (siehe auch CosH_1ea), dann MUSS als Trailer NoError verwendet werden.**[<=]**

G2_N086.000 - (N086.000) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoAuthentication gewählt werden.**[<=]**

A_16518 - (N086.100)a K_TST
~~Die Priorität der Trailer in CosT_599 MUSS herstellerspezifisch sein.~~**[<=]**

G2_N086.100.b - (N086.100)b K_COS
~~Jeder Trailer in CosT_599 MUSS eine höhere Priorität als NoError haben.~~**[<=]**

G2_N086.100.c - (N086.100)c K_COS
~~NoError MUSS eine höhere Priorität als NoAuthentication haben.~~**[<=]**

14.7.4 INTERNAL AUTHENTICATE

Das Kommando INTERNAL AUTHENTICATE berechnet Authentisierungsdaten zu einem Token mittels eines symmetrischen oder privaten Schlüssels. Der Schlüssel wird vor der Authentisierungsoperation ausgewählt. Dies geschieht vor dem Senden dieses INTERNAL AUTHENTICATE-Kommandos durch ein MSE-Set-Kommando (siehe (N100.400) und (N100.900)). Das Token ist als Parameter in der Kommandonachricht enthalten.

14.7.4.1 Use Case internes Authentisieren

In dieser Variante enthält die APDU des INTERNAL AUTHENTICATE-Kommandos zwei Parameter:

A_16584 - (N086.200) K_externeWelt {K_Karte}
Der Parameter *token* enthält die zu authentisierenden Daten. Der Parameter *token* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *token* MUSS abhängig von der mittels (N100.300) oder (N100.800) ausgewählten *algId* gewählt werden.**[<=]**

A_16585 - (N086.200)a K_externeWelt {K_Karte}
Wenn *algId* gleich eIcRoleAuthentication ist, dann MUSS *token* gleich 24 Oktett lang sein.**[<=]**

A_16586 - (N086.200)b K_externeWelt {K_Karte}
Wenn *algId* gleich rsaClientAuthentication ist, dann DARF *token* NICHT länger als 64 Oktette sein.**[<=]**

(N086.200)c ist absichtlich leer.
(N086.200)d ist absichtlich leer.

N086.200.c - (N086.200)c K_externeWelt {K_Karte} Option_RSA_CVC
~~Wenn *algId* gleich rsaRoleAuthentication ist, dann MUSS *token* gleich 16 Oktette lang sein.~~**[<=]**

N086.200.d - (N086.200)d K_externeWelt {K_Karte} Option_DES
~~Wenn *algId* gleich rsaSessionkey4SM ist, dann MUSS *token* gemäß (N106.700) gewählt werden.~~**[<=]**

A_16587 - (N086.200)e K_externeWelt {K_Karte}
Wenn *algId* gleich signPKCS1_V1_5 ist, dann MUSS die Anzahl Oktette in *token* kleiner als 0,4 OctetLength(*n*) sein, mit *n* als Modulus des ausgewählten Authentisierungsschlüssels.**[<=]**

A_16588 - (N086.202)a K_externeWelt {K_Karte}, Option_Kryptobox
Wenn *algId* gleich aesSessionkey4TC ist, dann MUSS *token* gleich 24 Oktett lang sein.**[<=]**

(N086.202)b ist absichtlich leer.
(N086.202)c ist absichtlich leer.

A_16589 - (N086.202)b K_externeWelt {K_Karte}, Option_Kryptobox, Option_DES
~~Wenn *algId* gleich desSessionkey4TC ist, dann MUSS *token* gleich 16 Oktett lang sein.~~**[<=]**

A_16590 - (N086.202)c K_externeWelt {K_Karte}, Option_Kryptobox, Option_DES
~~Wenn *algId* gleich rsaSessionkey4TC ist, dann MUSS *token* gleich 16 Oktett lang sein.~~**[<=]**

A_16591 - (N086.300) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *response* in der Antwortnachricht enthalten ist.**[<=]**

A_16592 - (N086.400) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_1d2 verwendet werden.

Tabelle 254: CosT_1d2: INTERNAL AUTHENTICATE

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'88'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Information zum Algorithmus bereits in der Karte vorhanden
P2	'00'	Schlüsselreferenz bereits in der Karte vorhanden
Data	'XX…XX'	*token*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.7.4.2 Antwort der Karte auf internes Authentisieren

Tabelle 255: CosT_710: INTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*response*	Authentisierende Daten
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	Erfolgreiche Authentisierungsoperation

Tabelle 256: CosT_235: INTERNAL AUTHENTICATE Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	KeyInvalid	Schlüsseldaten fehlen
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoKeyReference	Kein Authentisierungsschlüssel ausgewählt
~~'69 85'~~	~~NoPukReference~~	~~Kein Verschlüsselungsschlüssel ausgewählt~~
`6A 80'	WrongToken	fehlerhaftes Token
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 88'	KeyNotFound	Authentisierungsschlüssel nicht gefunden
~~'6A 88'~~	~~PukNotFound~~	~~Verschlüsselungsschlüssel nicht gefunden~~

Hinweis CosH_0aa: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16593 - (N086.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das INTERNAL AUTHENTICATE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.7.4.3 Kommandoabarbeitung innerhalb der Karte

G2_N086.600.a - (N086.600)a K_COS
Das COS MUSS die INTERNAL AUTHENTICATE-Variante aus
- 14.7.4.1- Use Case internes Authentisieren
unterstützen.**[<=]**

A_16594 - (N086.600)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere INTERNAL AUTHENTICATE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N086.700.a - (N086.700)a K_COS
Wenn *channelContext.keyReferenceList.internalAuthenticate* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16595 - (N086.700)b K_COS
Wenn *channelContext.keyReferenceList.internalAuthenticate* nicht leer ist, dann wird *affectedObject* = SearchKey(
currentFolder,

keyReferenceList.internalAuthenticate.keyReference,
keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N086.700.b.1 - (N086.700)b.1 K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N086.700.b.2 - (N086.700)b.2 K_COS
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N086.800 - (N086.800) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N086.810 - (N086.810) K_COS
Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.**[<=]**

G2_N086.820 - (N086.820) K_COS
Wenn *channelContext.keyReferenceList.internalAuthenticate.algID* einen Wert aus der Menge {aesSessionkey4TC, ~~desSessionkey4TC,~~ elcRoleAuthentication~~, rsaRoleAuthentication, rsaSessionkey4SM, rsaSessionkey4TC~~} besitzt und die acht LSByte von *token* identisch sind zu *iccsn8* (siehe (N019.900)c), genau dann MUSS das Kommando mit dem Trailer WrongToken terminieren.**[<=]**

A_16596 - (N086.900) K_COS
Die Antwort *response* MUSS wie folgt in Abhängigkeit von *algID* = *channelContext.keyReferenceList.internalAuthenticate.algID* berechnet werden:**[<=]**

G2_N086.900.a - (N086.900)a K_COS
Falls *algID* den Wert elcRoleAuthentication besitzt, dann gilt mit *PrK* = *affectedObject.privateKey*:
response = ELC_SIG(*PrK*, I2OS(OS2I(*token* || elcRoleAuthentication), *PrK.domainParameter. t* /8)).**[<=]**

G2_N086.900.b - (N086.900)b K_COS
Falls *algID* den Wert rsaClientAuthentication besitzt, dann gilt
response = RSASSA_PSS_SIGN(*PrK*, *token*).**[<=]**

(N086.900)c ist absichtlich leer.
(N086.900)d ist absichtlich leer.

A_16597 - (N086.900)c K_COS, Option_RSA_CVC
~~Falls *algID* den Wert , rsaRoleAuthentication besitzt, dann MÜSSEN mit der Definition *PrK = affectedObject.privateKey* folgende Schritte ausgeführt werden:~~**[<=]**

G2_N086.900.c.1 - (N086.900)c.1 K_COS, Option_RSA_CVC
~~Schritt c.1: PRND = RAND(OctetLength(*PrK.n*) – 34)~~**[<=]**

G2_N086.900.c.2 - (N086.900)c.2 K_COS, Option_RSA_CVC
~~Schritt c.2: *M* = PRND || *token*~~**[<=]**

G2_N086.900.c.3 - (N086.900)c.3 K_COS, Option_RSA_CVC

Schritt c.3: (*response*, *M2*) = RSA_ISO9796_2_DS1_SIGN(*PrK*, *M*)
Hinweis CosH_c2d: Die Länge von PRND ist so gewählt, dass M2 = token *ist. Darum ist es nicht notwendig,* M2 *in die Antwortnachricht einzustellen.*[<=]

A_16598 - (N086.900)d K_COS, Option_DES
Falls *algID* den Wert rsaSessionkey4SM besitzt, dann MÜSSEN mit der Definition *PrK = affectedObject.privateKey* folgende Schritte ausgeführt werden:[<=]

G2_N086.900.d.1 - (N086.900)d.1 K_COS, Option_DES
Schritt d.1: KD.i = RAND(64).
Der Oktettstring KD.i MUSS an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).[<=]

G2_N086.900.d.2 - (N086.900)d.2 K_COS, Option_DES
Schritt d.2: PRND = RAND(OctetLength(*PrK.n*) – 34 – OctetLength(KD.i))[<=]

G2_N086.900.d.3 - (N086.900)d.3 K_COS, Option_DES
Schritt d.3: *M* = PRND || KD.i || *token*[<=]

G2_N086.900.d.4 - (N086.900)d.4 K_COS, Option_DES
Schritt d.4: (*sig*, *M2*) = RSA_ISO9796_2_DS1_SIGN(*PrK*, *M*)
Hinweis CosH_c73: Die Länge von PRND ist so gewählt, dass M2 = token *ist. Darum ist es nicht notwendig, M2 in die Antwortnachricht einzustellen.*[<=]

G2_N086.900.d.5.i - (N086.900)d.5.i K_COS, Option_DES
Schritt d.5.i: Wenn *channelContext.keyReferenceList.externalAuthenticate* leer ist, genau dann MUSS das Kommando mit dem Trailer NoPukReference terminieren.[<=]

A_16599 - (N086.900)d.5.ii K_COS, Option_DES
Schritt d.5.ii: Wenn *channelContext.keyReferenceList.externalAuthenticate* nicht leer, dann wird
tmpObject = SearchKey(
currentFolder,
keyReferenceList.externalAuthenticate.keyReference,
keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß CosT_995 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Wenn die Schlüsselsuche den Fehler
A. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PukNotFound terminieren.
B. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.[<=]

G2_N086.900.d.6 - (N086.900)d.6 K_COS, Option_DES
Schritt d.6: *response = sig*$^{PuK.e}$ mod *PuK.n*, mit *PuK = tmpObject.publicKey*.[<=]

G2_N086.900.e - (N086.900)e K_COS
Falls *algID* den Wert signPKCS1_V1_5 besitzt, dann gilt:
response = RSASSA_PKCS1_V1_5_SIGN(*PrK*, *token*)[<=]

A_16600 - (N086.902)a K_COS, Option_Kryptobox
Falls *channelContext.keyReferenceList.internalAuthenticate.algID* den Wert aesSessionkey4TC besitzt, dann MÜSSEN folgende Schritte ausgeführt werden:[<=]

G2_N086.902.a.1 - (N086.902)a.1 K_COS, Option_Kryptobox
Schritt a.1: Setze RND.int = RAND(16)[<=]

G2_N086.902.a.2 - (N086.902)a.2 K_COS, Option_Kryptobox

Schritt a.2: Setze ICCSN8.int = *iccsn8* (siehe (N019.900)c)**[<=]**

G2_N086.902.a.3 - (N086.902)a.3 K_COS, Option_Kryptobox
Schritt a.3: Setze KD.i = RAND(64)**[<=]**

G2_N086.902.a.4 - (N086.902)a.4 K_COS, Option_Kryptobox
Schritt a.4: Setze S = RND.int || ICCSN8.int || *token* || KD.i**[<=]**

G2_N086.902.a.5 - (N086.902)a.5 K_COS, Option_Kryptobox
Schritt a.5: Setze *C*1 = AES_CBC_ENC(*affectedObject.encKey*, 0, S) **[<=]**

G2_N086.902.a.6 - (N086.902)a.6 K_COS, Option_Kryptobox
Schritt a.6: Setze *MAC*1 = CalculateCMAC_IsoPadding(*affectedObject.macKey*, *C*1)**[<=]**

G2_N086.902.a.7 - (N086.902)a.7 K_COS, Option_Kryptobox
Schritt a.7: Setze *response* = *C*1 || *MAC*1 **[<=]**

G2_N086.902.a.8 - (N086.902)a.8 K_COS, Option_Kryptobox
Schritt a.8: RND.int MUSS als *RND.ICC* gespeichert werden (siehe (N029.900)b).**[<=]**

G2_N086.902.a.9 - (N086.902)a.9 K_COS, Option_Kryptobox
Schritt a.9: KD.i MUSS an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).**[<=]**

(N086.902)b ist absichtlich leer.
(N086.902)c ist absichtlich leer.

A_16601 - (N086.902)b K_COS, Option_DES
~~Falls *channelContext.keyReferenceList.internalAuthenticate.algID* den Wert desSessionkey4TC besitzt, dann MÜSSEN folgende Schritte ausgeführt werden:~~**[<=]**

G2_N086.902.b.1 - (N086.902)b.1 K_COS, Option_DES
~~Schritt b.1: Setze RND.int = RAND(8).~~**[<=]**

G2_N086.902.b.2 - (N086.902)b.2 K_COS, Option_DES
~~Schritt b.2: Setze ICCSN8.int = *iccsn8* (siehe (N019.900)c)~~**[<=]**

G2_N086.902.b.3 - (N086.902)b.3 K_COS, Option_DES
~~Schritt b.3: Setze KD.i = RAND(64).~~**[<=]**

G2_N086.902.b.4 - (N086.902)b.4 K_COS, Option_DES
~~Schritt b.4: Setze S = RND.int || ICCSN8.int || *token* || KD.i~~**[<=]**

G2_N086.902.b.5 - (N086.902)b.5 K_COS, Option_DES
~~Schritt b.5: Setze *C*1 = 3TDES_CBC_ENC(*affectedObject.encKey*, 0, S)~~**[<=]**

G2_N086.902.b.6 - (N086.902)b.6 K_COS, Option_DES
~~Schritt b.6: Setze *MAC*1 = CALCULATE_Retail_MAC(*affectedObject.macKey*, *C*1)~~**[<=]**

G2_N086.902.b.7 - (N086.902)b.7 K_COS, Option_DES
~~Schritt b.7: Setze *response* = *C*1 || *MAC*1~~**[<=]**

G2_N086.902.b.8 - (N086.902)b.8 K_COS, Option_DES
~~Schritt b.8: RND.int MUSS als *RND.ICC* gespeichert werden (siehe (N029.900)b).~~**[<=]**

G2_N086.902.b.9 - (N086.902)b.9 K_COS, Option_DES
~~Schritt b.9: KD.i MUSS an den Secure Messaging Layer übergeben werden (siehe CosT_a2e).~~**[<=]**

G2_N086.902.c - (N086.902)c K_COS, Option_DES

Falls *channelContext.keyReferenceList.internalAuthenticate.algID* den Wert rsaSessionkey4TC besitzt, dann MUSS im Rahmen der Kommandobearbeitung identisch zu rsaSessionkey4SM verfahren werden.**[<=]**

G2_N087.000 - (N087.000) K_COS
Als Datenfeld der Antwortnachricht MUSS *response* verwendet werden.**[<=]**

G2_N087.100 - (N087.100) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16602 - (N087.200)a K_TST
Die Priorität der Trailer in CosT_235 MUSS herstellerspezifisch sein.**[<=]**

G2_N087.200.b - (N087.200)b K_COS
Jeder Trailer in CosT_235 MUSS eine höhere Priorität als NoError haben.**[<=]**

Hinweis 125: Bei der Schlüsselsuche in (N086.900)d.5.ii wird nach einem öffentlichen Schlüsselobjekt gesucht. Trotzdem wird algID *aus* keyReferenceList.internalAuthenticate *entnommen, damit sicher gestellt ist, dass auch das öffentliche Schlüsselobjekt den hier zum Tragen kommenden Algorithmus unterstützt. Aus der übrigen Spezifikation ergibt sich, dass dieser Algorithmus Element von {rsaSessionkey4SM, rsaSessionkey4TC} ist.*

14.8 Kryptographische Operationen

Hinweis CosH_b01: In einer früheren Version trug dieses Kapitel die Überschrift "Kryptoboxkommandos". Es wurde umbenannt in "Kryptographische Operationen", um Verwechselungen mit der Option_Kryptobox zu vermeiden.

Dieses Unterkapitel behandelt Funktionalitäten, die in [ISO/IEC 7816-8] mit dem Kommando PERFORM SECURITY OPERATION verknüpft und alle über den INS Code '2A' erreichbar sind. Die folgende Tabelle gibt einen informativen Überblick über die hier behandelten Funktionalitäten. Einzelheiten finden sich in den nachfolgenden Kapiteln.

Tabelle 257: CosT_c98: Tabelle aller PERFORM SECURITY OPERATION Kommando Header

CLA	INS	P1	P2	Kommando	Referenz
'00'	'2A'	'8E'	'80'	PSO Compute Cryptographic Checksum	14.8.1
		'9E'	'9A'	PSO Compute Digital Signature, ohne "recovery"	14.8.2.1
		'9E'	'AC'	PSO Compute Digital Signature, mit "recovery"	14.8.2.2
		'80'	'86'	PSO Decipher	14.8.3
		'86'	'80'	PSO Encipher	14.8.4
		'90'	'XX'	PSO Hash	14.8.5
		'86'	'B8'	PSO Transcipher mittels RSA	14.8.6
		'00'	'BE'	PSO Verify Certificate	14.8.7.2
		'00'	'A2'	PSO Verify Cryptographic Checksum	14.8.8
		'00'	'A8'	PSO Verify Digital Signature	14.8.9

14.8.1 PSO Compute Cryptographic Checksum

Das Kommando PSO Compute Cryptographic Checksum berechnet zu gegebenen Daten eine kryptographische Checksumme mittels eines symmetrischen Schlüssels. Der symmetrische Schlüssel wird im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2~~, 15.4.3~~ oder 15.4.4) ausgehandelt. Die durch eine Checksumme zu schützenden Daten sind als Parameter in der Kommandonachricht enthalten.

14.8.1.1 Use Case Berechnen einer kryptographischen Checksumme

Diese Variante gilt für Algorithmen aus der folgenden Menge: {aesSessionkey~~, desSessionkey (Option_DES)~~}.

In dieser Variante enthält die APDU des PSO Compute Cryptographic Checksum Kommandos drei Parameter:

A_16603 - (N087.220)a K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *flagSSCmacIncrement* MUSS wie folgt gewählt werden:**[<=]**

(N087.220)a.1 ist absichtlich leer.

N07.220.a.1 - (N087.220)a.1 K_externeWelt {K_Karte}, (Option_Kryptobox oder Option_PACE_PCD) und Option_DES
~~Wenn es sich um Sessionkeys für den Algorithmus desSessionkey handelt, dann ist *flagSSCmacIncrement* ein leerer Oktettstring.~~**[<=]**

A_16604 - (N087.220)a.2 K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Wenn es sich um Sessionkeys für den Algorithmus aesSessionkey handelt und *SSCmac*
i. zu inkrementieren ist, dann ist *flagSSCmacIncrement* = '01'.
ii. unverändert zu verwenden ist, dann ist *flagSSCmacIncrement* = '00'.**[<=]**

A_16605 - (N087.220)b K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *data* enthält die zu schützenden Daten. Der Parameter *data* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *data* MUSS aus dem in (N026.900) definierten Bereich gewählt werden.**[<=]**

A_16606 - (N087.220)c K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {WildCardShort, WildCardExtended} gewählt werden.**[<=]**

A_16607 - (N087.228) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_861 verwendet werden.

Tabelle 258: CosT_861: PSO Compute Cryptographic Checksum, Berechnen eines MAC

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'8E'	Beschreibung der Antwortdaten, hier kryptographische Checksumme
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX…XX'	*flagSSCmacIncrement* || *data*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.1.2 Antwort der Karte auf Berechnen einer kryptographischen Checksumme

Tabelle 259: CosT_7e7: PSO Compute Cryptographic Checksum Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*mac*	kryptographische Checksumme
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	erfolgreiche Berechnung eines MAC

Tabelle 260: CosT_532: PSO Compute Cryptographic Checksum Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoKeyReference	kein Schlüssel für MAC Berechnung ausgewählt
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 88'	KeyNotFound	kein Schlüssel für MAC Berechnung vorhanden

Hinweis CosH_7cb: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16608 - (N087.232) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Compute Cryptographic Checksum-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.1.3 Kommandoabarbeitung innerhalb der Karte

G2_N087.236.a - (N087.236)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Das COS MUSS die PSO Compute Cryptographic Checksum Variante aus
- 14.8.1.1- Use Case Berechnen einer kryptographischen Checksumme

unterstützen.**[<=]**

A_16609 - (N087.236)b K_TST

~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Compute Cryptographic Checksum-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N087.240.a - (N087.240)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *channelContext.keyReferenceList.macCalculation* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16610 - (N087.240)b K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *channelContext.keyReferenceList.macCalculation* nicht leer ist, dann wird *affectedObject* = SearchKey(
currentFolder,
keyReferenceList.macCalculation.keyReference,
keyReferenceList.macCalculation.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N087.240.b.1 - (N087.240)b.1 K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N087.240.b.2 - (N087.240)b.2 K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N087.244 - (N087.244) K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

Mit den Attributen *SSCmac* und *Kmac* aus dem Attribut *SessionkeyContext* gilt:

G2_N087.248.a - (N087.248)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *keyReferenceList.macCalculation.algID* den Wert aesSessionkey besitzt, MUSS gelten:
1. Wenn *flagSSCmacIncrement* = '01' ist,
dann wird *SSCmac* inkrementiert: *SSCmac* = *SSCmac* + 1,
sonst bleibt *SSCmac* unverändert.
2. *mac* = CalculateCMAC_IsoPadding(*Kmac*, I2OS(*SSCmac*, 16) || *data*).**[<=]**

(N087.248)b ist absichtlich leer.

G2_N087.248.b - (N087.248)b K_COS, (Option_Kryptobox oder Option_PACE_PCD) und Option_DES
~~Wenn *keyReferenceList.macCalculation.algID* den Wert desSessionkey besitzt, MUSS gelten:~~
~~1. *SSCmac* = *SSCmac* + 1.~~
~~2. *mac* = CALCULATE_Retail_MAC(*Kmac*, I2OS(*SSCmac*, 8) || *data*).~~**[<=]**

G2_N087.252 - (N087.252) K_COS, Option_Kryptobox oder Option_PACE_PCD
Als Datenfeld der Antwortnachricht MUSS *mac* verwendet werden.**[<=]**

G2_N087.256 - (N087.256) K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16611 - (N087.260)a K_TST, Option_Kryptobox oder Option_PACE_PCD
Die Priorität der Trailer in CosT_532 MUSS herstellerspezifisch sein.**[<=]**

G2_N087.260.b - (N087.260)b K_COS, Option_Kryptobox oder Option_PACE_PCD
Jeder Trailer in CosT_532 MUSS eine höhere Priorität als NoError haben.**[<=]**

14.8.2 PSO Compute Digital Signature

Das Kommando PSO Compute Digital Signature signiert Daten mittels eines privaten Schlüssels. Der private Schlüssel und der zu verwendende Algorithmus werden vor der Signieroperation ausgewählt. Dies geschieht vor dem Senden dieses PSO Compute Digital Signature-Kommandos durch ein MSE-Set-Kommando (siehe (N102.900)). Die zu signierenden Daten sind als Parameter in der Kommandonachricht enthalten.

14.8.2.1 Use Case Signieren des Datenfeldes, ohne "message recovery"

Diese Variante gilt für folgende Algorithmen (siehe (N017.600) und (N018.300)): {rsaClientAuthentication, signECDSA, signPKCS1_V1_5, signPSS}.

In dieser Variante enthält die APDU des PSO Compute Digital Signature-Kommandos zwei Parameter:

A_16612 - (N087.300) K_externeWelt {K_Karte}
Der Parameter *dataToBeSigned* enthält die zu signierenden Daten. Der Parameter *dataToBeSigned* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *dataToBeSigned* MUSS in Abhängigkeit von der mittels (N102.800) ausgewählten *algId* gewählt werden.**[<=]**

A_16613 - (N087.300)a K_externeWelt {K_Karte}
Wenn *algId* gleich rsaClientAuthentication ist, dann MUSS die Länge von *dataToBeSigned* kleiner gleich 64 Oktette sein.**[<=]**

A_16614 - (N087.300)b K_externeWelt {K_Karte}
Wenn *algId* gleich signECDSA ist, dann MUSS die Länge von *dataToBeSigned* gleich dem Parameter *domainParameter*. *t* des Signierschlüssels sein.**[<=]**

A_16615 - (N087.300)c K_externeWelt {K_Karte}
Wenn *algId* gleich signPKCS1_V1_5 ist, dann MUSS die Länge von *dataToBeSigned* kleiner als 0,4 OctetLength(*n*) sein, mit *n* als Modulus des ausgewählten Signaturschlüssels.**[<=]**

A_16616 - (N087.300)d K_externeWelt {K_Karte}
Wenn *algId* gleich signPSS ist, dann MUSS die Länge von *dataToBeSigned* kleiner gleich 64 Oktette sein.**[<=]**

A_16617 - (N087.400) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *signature* in der Antwortnachricht enthalten ist.**[<=]**

A_16618 - (N087.500) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_c48 verwendet werden.

Tabelle 261: CosT_c48: PSO Compute Digital Signature, Signieren ohne "message recovery"

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'9E'	Beschreibung der Antwortdaten, hier digitale Signature
P2	'9A'	Beschreibung der Kommandodaten, hier "zu signierende Daten"
Data	'XX…XX'	*dataToBeSigned*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.2.2 Use Case Signieren des Datenfeldes, mit "message recovery"

Diese Variante gilt für folgenden Algorithmus (siehe (N018.300)): {sign9796_2_DS2}.

In dieser Variante enthält die APDU des PSO Compute Digital Signature-Kommandos drei Parameter:

A_16619 - (N087.600) K_externeWelt {K_Karte}
Der Parameter *M1* enthält den "recoverable part" der zu signierenden Nachricht. Der Parameter*M1* ist ein Oktettstring mit beliebigem Inhalt. Die Bitlänge von *M1* DARF die Kapazität des Schemas gemäß [ISO/IEC 9796-2#9.2.4] NICHT überschreiten.**[<=]**

Hinweis CosH_7cb: Die Kapazität eines RSA-Schlüssels (siehe (N002.100)) beträgt für die Moduluslänge
*a. 2048 bit: c = k – L_h – L_s – 8t – 2 = (2048 – 256 – 256 – 8*1 – 2) bit = 1526 bit.*
*b. 3072 bit: c = k – L_h – L_s – 8t – 2 = (3072 – 256 – 256 – 8*1 – 2) bit = 2550 bit.*

A_16620 - (N087.700) K_externeWelt {K_Karte}
Der Parameter *hashM2* enthält den Hash-Wert des "non recoverable part" der zu signierenden Nachricht. Der Parameter*hashM2* ist ein Oktettstring mit beliebigem Inhalt, dessen Länge kleiner gleich 64 sein MUSS.**[<=]**

A_16621 - (N087.800) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *signature* in der Antwortnachricht enthalten ist.**[<=]**

A_16622 - (N087.900) K_externeWelt {K_Karte}
Die Parameter *M1* und *hashM2* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N088.600)b spezifiziert. Die Anzahl der Oktette in *signInput9796_2_DS2* wird durch (N087.600) und (N087.700) beschränkt.**[<=]**

A_16623 - (N088.000) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_f84 verwendet werden.

Tabelle 262: CosT_f84: PSO Compute Digital Signature, Signieren "message recovery"

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'9E'	Beschreibung der Antwortdaten, hier digitale Signature
P2	'AC'	Beschreibung der Kommandodaten, hier zu signierende Daten als DE
Data	'XX…XX'	*signInput9796_2_DS2*, DER codiertes Datenfeld
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.2.3 Antwort der Karte auf Signieren von Daten

Tabelle 263: CosT_562: PSO Compute Digital Signature Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*signature*	Signatur
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	Erfolgreiche Signaturoperation

Tabelle 264: CosT_7c1: PSO Compute Digital Signature Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	KeyInvalid	Schlüsseldaten fehlen
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoKeyReference	Kein Signierschlüssel ausgewählt
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den geforderten Algorithmus nicht
'6A 88'	KeyNotFound	Schlüssel nicht gefunden

Hinweis CosH_926: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16624 - (N088.100) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Compute Digital Signature-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.2.4 Kommandoabarbeitung innerhalb der Karte

G2_N088.200.a - (N088.200)a K_COS
Das COS MUSS die PSO Compute Digital Signature-Varianten aus
- 14.8.2.1- Use Case Signieren des Datenfeldes, ohne "message recovery" und
- 14.8.2.2- Use Case Signieren des Datenfeldes, mit "message recovery"

unterstützen.**[<=]**

A_16625 - (N088.200)b K_TST

~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Compute Digital Signature-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N088.300.a - (N088.300)a K_COS
Wenn *channelContext.keyReferenceList.signatureCreation* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16626 - (N088.300)b K_COS
Wenn *channelContext.keyReferenceList.signatureCreation* nicht leer ist, dann wird *affectedObject* = SearchKey(
currentFolder,
keyReferenceList.signatureCreation.keyReference,
keyReferenceList.signatureCreation.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N088.300.b.1 - (N088.300)b.1 K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N088.300.b.2 - (N088.300)b.2 K_COS
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N088.400 - (N088.400) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N088.500 - (N088.500) K_COS
Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.**[<=]**

Die Signatur *signature* wird in Abhängigkeit der gewählten *algID* berechnet.

G2_N088.600.a - (N088.600)a K_COS
Wenn *keyReferenceList.signatureCreation.algID* den Wert rsaClientAuthentication oder signPSS besitzt, dann gilt:
signature = RSASSA_PSS_SIGN(*affectedObject.privateRsaKey*, *dataToBeSigned*).**[<=]**

G2_N088.600.b - (N088.600)b K_COS
Wenn *keyReferenceList.signatureCreation.algID* den Wert sign9796_2_DS2 besitzt, dann gilt:
1. *signInput9796_2_DS2* == *plainDO* || *hashDO*.
2. *plainDO* == '80 – L_{80} – *M1* '.
3. *hashDO* == '90 – L_{90} – *hashM2* '.
4. *signature* = RSA_ISO9796_2_DS2_SIGN(*affectedObject.privateRsaKey*, *M1*, *hashM2*).**[<=]**

G2_N088.600.c - (N088.600)c K_COS
Wenn *keyReferenceList.signatureCreation.algID* den Wert signECDSA besitzt, dann gilt:
1. (*R*, *S*) = ELC_SIG(*affectedObject.privateElcKey*, *dataToBeSigned*).
2. *signature* = *R* || *S*. **[<=]**

G2_N088.600.d - (N088.600)d K_COS

Wenn *keyReferenceList.signatureCreation.algID* den Wert signPKCS1_V1_5 besitzt, dann gilt:
signature = RSASSA_PKCS1_V1_5_SIGN(*affectedObject.privateRsaKey*, *dataToBeSigned*).**[<=]**

G2_N088.700 - (N088.700) K_COS
Als Datenfeld der Antwortnachricht MUSS *signature* verwendet werden.**[<=]**

G2_N088.800 - (N088.800) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16627 - (N088.900)a K_TST
Die Priorität der Trailer in CosT_7c1 MUSS herstellerspezifisch sein.**[<=]**

G2_N088.900.b - (N088.900)b K_COS
Jeder Trailer in CosT_7c1 MUSS eine höhere Priorität als NoError haben.**[<=]**

14.8.3 PSO Decipher

Das Kommando PSO Decipher entschlüsselt Daten, die als Parameter in der Kommandonachricht enthalten sind. Für den Entschlüsselungsschlüssel und den zu verwendenden Algorithmus gibt es mehrere Optionen

- Wird ein symmetrischer Schlüssel verwendet, so wird dieser im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2, ~~15.4.3~~ oder 15.4.4) ausgehandelt und zusammen mit einem Algorithmus implizit ausgewählt.
- Wird ein privater Schlüssel verwendet, der persistent in der Smartcard gespeichert ist, so wird dieser Schlüssel und der zu verwendende Algorithmus vor dem Senden dieses PSO Decipher-Kommandos durch ein explizites MSE-Set-Kommando ausgewählt (siehe (N103.800)).

14.8.3.1 Use Case Entschlüsseln mittels RSA

Diese Variante gilt für Algorithmen (siehe (N017.900)a.1) aus folgender Menge: {~~rsaDecipherPKCS1_V1_5,~~ rsaDecipherOaep}.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos zwei Parameter:

A_16628 - (N089.000) K_externeWelt {K_Karte}
Der Parameter *C* enthält die zu entschlüsselnden Daten. Der Parameter *C* ist ein Oktettstring mit beliebigem Inhalt. Die Anzahl Oktette in *C* MUSS identisch sein zu OctetLength(*n*) mit *n* als Modulus des Schlüssels, der zum Entschlüsseln verwendet wird.**[<=]**

A_16629 - (N089.100) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.**[<=]**

A_16630 - (N089.200) K_externeWelt {K_Karte}
Es MUSS eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_643 verwendet werden.

Tabelle 265: CosT_643: PSO Decipher, Entschlüsseln mittels RSA

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'80'	Beschreibung der Antwortdaten, hier Klartext
P2	'86'	Beschreibung der Kommandodaten, hier Chiffrat
Data	'XX…XX'	'00' || C, dies bedeutet: PaddingIndicator || Cryptogram
Le	'0000'	$length$, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.3.2 Use Case Entschlüsseln mittels ELC

Diese Variante gilt für Algorithmen (siehe (N017.900)a.2) aus folgender Menge: {elcSharedSecretCalculation}.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos vier Parameter:

A_16631 - (N089.300) K_externeWelt {K_Karte}
Der Parameter PO_A enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt wurde vom Sender der Nachricht gewählt. Der Parameter PO_A ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).**[<=]**

A_16632 - (N089.400) K_externeWelt {K_Karte}
Der Parameter C enthält die zu entschlüsselnden Daten. Der Parameter C MUSS ein Oktettstring mit beliebigem Inhalt sein.**[<=]**

A_16633 - (N089.500) K_externeWelt {K_Karte}
Der Parameter T enthält einen MAC, der die Integrität von C schützt. Der Parameter T ist ein Oktettstring, dessen Länge und Inhalt so gewählt werden SOLL, dass bei der MAC-Prüfung kein Fehler auftritt.**[<=]**

A_16634 - (N089.600) K_externeWelt {K_Karte}
Der Parameter $length$ bestimmt die Länge der erwarteten Antwortdaten. Der Wert von $length$ MUSS aus der Menge {WildCardShort, WildCardExtended} gewählt werden.**[<=]**

A_16635 - (N089.700) K_externeWelt {K_Karte}
Die Parameter PO_A, C und T MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung MUSS den Vorgaben in (N090.300)c entsprechen.**[<=]**

A_16636 - (N089.800) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_767 verwendet werden.

Tabelle 266: CosT_767: PSO Decipher, Entschlüsseln mittels elliptischer Kurven

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'80'	Beschreibung der Antwortdaten, hier Klartext
P2	'86'	Beschreibung der Kommandodaten, hier Chiffrat
Data	'XX…XX'	*cipher*, DER codiertes Datenfeld
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

~~*Hinweis CosH_e00(129) Möglicherweise wird P2 in einer späteren Dokumentenversion geändert.*~~

14.8.3.3 Use Case Entschlüsseln mittels symmetrischer Schlüssel

Diese Variante gilt für Algorithmen aus der folgenden Menge: {aesSessionkey~~, desSessionkey (Option_DES)~~}.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos zwei Parameter:

A_16637 - (N089.840) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter C enthält die zu entschlüsselnden Daten. Der Parameter C ist ein Oktettstring mit beliebigem Inhalt. Die Anzahl Oktette in C MUSS ein ganzzahliges Vielfaches der Blocklänge des verwendeten Kryptoalgorithmus sein.**[<=]**

A_16638 - (N089.843) K_externeWelt {K_ Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {WildCardShort, WildCardExtended} gewählt werden.**[<=]**

A_16639 - (N089.845) K_externeWelt {K_ Karte}, Option_Kryptobox oder Option_PACE_PCD
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_0f8 verwendet werden.

Tabelle 267: CosT_0f8: PSO Decipher, Entschlüsseln mittels symmetrischem Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'80'	Beschreibung der Antwortdaten, hier Klartext
P2	'86'	Beschreibung der Kommandodaten, hier Chiffrat
Data	'XX…XX'	'01' || C, dies bedeutet:Paddingindikator || Kryptogramm
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.3.4 Antwort der Karte auf Entschlüsseln von Daten

Tabelle 268: CosT_add: PSO Decipher Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*plain*	Klartext
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	Erfolgreiche Entschlüsselungsoperation

Tabelle 269: CosT_dbf: PSO Decipher Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	KeyInvalid	Schlüsseldaten fehlen
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoKeyReference	Kein Schlüssel für Entschlüsselung ausgewählt
'6A 80'	WrongCiphertext	Fehler beim Entschlüsseln des Chiffrats
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 88'	KeyNotFound	Schlüssel nicht gefunden

Hinweis CosH_d87: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16640 - (N089.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Decipher-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.3.5 Kommandoabarbeitung innerhalb der Karte

G2_N090.000.a - (N090.000)a K_COS
Das COS MUSS die PSO Decipher-Varianten aus
- 14.8.3.1- Use Case Entschlüsseln mittels RSA und
- 14.8.3.2- Use Case Entschlüsseln mittels ELC
unterstützen.**[<=]**

G2_N090.000.b - (N090.000)b K_COS, Option_Kryptobox oder Option_PACE_PCD
Das COS MUSS die PSO Decipher-Variante aus
- 14.8.3.3- Use Case Entschlüsseln mittels symmetrischer Schlüssel
unterstützen.**[<=]**

A_16641 - (N090.000)c K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Decipher-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N090.100.a - (N090.100)a K_COS

Wenn *channelContext.keyReferenceList.dataDecipher* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16642 - (N090.100)b K_COS
Wenn *channelContext.keyReferenceList.dataDecipher* nicht leer ist, dann wird *affectedObject* = SearchKey(
currentFolder,
keyReferenceList.dataDecipher.keyReference,
keyReferenceList.dataDecipher.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N090.100.b.1 - (N090.100)b.1 K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N090.100.b.2 - (N090.100)b.2 K_COS
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N090.200 - (N090.200) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N090.210 - (N090.210) K_COS
Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.**[<=]**

Die Klartextnachricht *plain* wird wie folgt berechnet:

(N090.300) a Diese Anforderung ist absichtlich leer.

G2_N090.300.b - (N090.300)b K_COS
Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert rsaDecipherOaep besitzt, dann gilt
plain = RSAES_OAEP_DECRYPT(*affectedObject.privateRsaKey*, *C*) **[<=]**

G2_N090.300.c - (N090.300)c K_COS
Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert elcSharedSecretCalculation besitzt, dann gilt
(*Hinweis CosH_7de:* cipher *ist hier identisch zu (N085.068)b.7, (N091.700)d und (N094.400)c definiert*):
1. *cipher* == 'A6 – L_{A6} – (*oidDO* || *keyDO* || *cipherDO* || *macDO*).
2. *oidDO* == '06 – L_{06} – *oid* '.
3. *keyDO* == '7F49 – L_{7F49} – (86 – L_{86} – PO_A)'.
4. *cipherDO* == '86 – L_{86} – (02 || *C*)'.
5. *macDO* == '8E – L_{8E} – *T* '.
6. Wenn *oid* verschieden ist zur OID, die gemäß (N008.600)d zu *affectedObject. privateElcKey. domainParameter* gehört, dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren. Diese Domainparameter werden im Folgenden mit *dP* bezeichnet.
7. *plain* = ELC_DEC(OS2P(PO_A, *dP*), *affectedObject.privateElcKey*, *C*, *T*).**[<=]**

G2_N090.302.a - (N090.302)a K_COS, Option_Kryptobox oder Option_PACE_PCD

Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert aesSessionkey besitzt, dann gilt mit den Attributen *SSCmac* und *Kenc* aus dem Attribut *SessionkeyContext*:
1. Schritt 1: *SSCenc* = OS2I(AES_ENC(K_{enc}, I2OS(*SSCmac*, 16))).
2. Schritt 2: *P* = AES_CBC_DEC(*Kenc*, *SSCenc*, *C*).
3. Schritt 3: *plain* = TruncateIso(*P*, 16).
4. Wenn die Funktion TruncateIso mit dem Fehler paddingError terminiert, dann
   i. MUSS das Kommando mit dem Trailer WrongCiphertext terminieren und
   ii. die Sessionkeys MÜSSEN mittels clearSessionkeys( ) gelöscht werden.**[<=]**

(N090.302)b ist absichtlich leer.

G2_N090.302.b - (N090.302)b K_COS, Option_DES und (Option_Kryptobox oder Option_PACE_PCD)
~~Falls *channelContext.keyReferenceList.dataDecipher.algID* den Wert desSessionkey besitzt, dann gilt mit den Attributen *SSCenc* und *Kenc* aus dem Attribut *SessionkeyContext*:~~
~~1. Schritt 1: *SSCenc* = *SSCenc* + 1.~~
~~2. Schritt 2: *P* = 3TDES_CBC_DEC(*Kenc*, *SSCenc*, *C*).~~
~~3. Schritt 3: *plain* = TruncateIso(*P*, 8).~~
~~4. Falls die Funktion TruncateIso mit dem Fehler paddingError terminiert, dann~~
~~i. MUSS das Kommando mit dem Trailer WrongCiphertext terminieren und~~
~~ii. die Sessionkeys MÜSSEN mittels clearSessionkeys( ) gelöscht werden.~~**[<=]**

G2_N090.400 - (N090.400) K_COS
Wenn die Entschlüsselung gemäß (N090.300) mit dem Fehler "*ERROR*" terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.**[<=]**

G2_N090.500 - (N090.500) K_COS
Als Datenfeld der Antwortnachricht MUSS *plain* verwendet werden.**[<=]**

G2_N090.600 - (N090.600) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16643 - (N090.700)a K_TST
Die Priorität der Trailer in CosT_dbf MUSS herstellerspezifisch sein.**[<=]**

G2_N090.700.b - (N090.700)b K_COS
Jeder Trailer in CosT_dbf MUSS eine höhere Priorität als NoError haben.**[<=]**

14.8.4 PSO Encipher

Das Kommando PSO Encipher verschlüsselt Daten, die als Parameter in der Kommandonachricht enthalten sind. Für den Verschlüsselungsschlüssel und den zu verwendenden Verschlüsselungsalgorithmus gibt es mehrere Optionen:

- Wird ein symmetrischer Schlüssel verwendet, so wird dieser im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2, 15.4.3 oder 15.4.4) ausgehandelt und zusammen mit einem Verschlüsselungsalgorithmus implizit ausgewählt.
- Wird ein öffentlicher Schlüssel verwendet, der persistent in der Smartcard gespeichert ist, so wird dieser Schlüssel und der zu verwendende Algorithmus vor dem Senden dieses PSO Encipher-Kommandos durch ein explizites MSE-Set-Kommando ausgewählt (siehe (N103.850)).

- Wird ein öffentlicher Schlüssel verwendet, der als Parameter in der Kommandonachricht enthalten ist, so ist auch der zu verwendende Algorithmus als Parameter in der Kommandonachricht enthalten.

14.8.4.1 Use Case Verschlüsseln von Daten mittels übergebenem RSA-Schlüssel

In dieser Variante wird der Schlüssel als Parameter in der Kommandonachricht übergeben. Folgende Algorithmen sind zulässig: {rsaEncipherOaep}.

Hinweis CosH_672: Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher *leer ist.*

In dieser Variante enthält die APDU des PSO Encipher-Kommandos vier Parameter:

A_16644 - (N090.780) K_externeWelt {K_Karte}
Der Parameter *algID* enthält Informationen zum Algorithmus, der für die Verschlüsselung verwendet wird. Der Parameter *algID* MUSS aus der Menge {rsaEncipherOaep} gewählt werden.**[<=]**

A_16645 - (N090.782) K_externeWelt {K_Karte}
Der Parameter *PuK* enthält den öffentlichen Schlüssel des Empfängers gemäß 8.2.4.1. Der Parameter *PuK* ist ein Oktettstring, dessen Inhalt so gewählt werden MUSS, dass bei der Decodierung kein Fehler auftritt (siehe (N091.700)c.3).**[<=]**

A_18354 - (N090.784) K_externeWelt {K_Karte}
Der Parameter M enthält die zu verschlüsselnden Daten. Der Parameter M ist ein Oktettstring mit beliebigem Inhalt. Die maximal mögliche Länge von M ist abhängig von der gemäß (N090.780) übergebenen *algId* und der Länge des Modulus n. **[<=]**

A_16646 - (N090.784)a K_externeWelt {K_Karte}
Wenn *algId* gleich rsaEncipherOaep ist, dann MUSS OctetLength(M) kleiner gleich OctetLength(n) – 66 sein.**[<=]**

(N090.784)b Diese Anforderung ist absichtlich leer.

A_16648 - (N090.786) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.**[<=]**

A_16649 - (N090.788) K_externeWelt {K_Karte}
Die Parameter *PuK*, *algID* und M MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N091.700)c spezifiziert.**[<=]**

A_16650 - (N090.790) K_externeWelt {K_Karte}
Es MUSS eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_da1 verwendet werden.

Tabelle 270: CosT_da1: PSO Encipher, Verschlüsseln mittels übergebenem RSA-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX…XX'	*plainDO*, DER codiertes Datenfeld
Le	'0000'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.4.2 Use Case Verschlüsseln von Daten mittels übergebenem ELC-Schlüssel

In dieser Variante wird der Schlüssel als Parameter in der Kommandonachricht übergeben. Folgende Algorithmen sind zulässig: {elcSharedSecretCalculation}.

Hinweis CosH_cb0: Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher *leer ist.*

In dieser Variante enthält die APDU des PSO Encipher-Kommandos fünf Parameter:

A_16651 - (N090.800) K_externeWelt {K_Karte}
Der Parameter *algID* enthält Informationen zum Algorithmus, der für die Verschlüsselung verwendet wird. Der Parameter *algID* MUSS aus der Menge {elcSharedSecretCalculation} gewählt werden.**[<=]**

A_16652 - (N090.900) K_externeWelt {K_Karte}
Der Parameter *oid* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve referenziert. Der Parameter *oid* MUSS aus der in CosT_a91 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.**[<=]**

A_16653 - (N091.000) K_externeWelt {K_Karte}
Der Parameter PO_B enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Empfängers. Der Parameter PO_B ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.500)a).**[<=]**

A_16654 - (N091.100) K_externeWelt {K_Karte}
Der Parameter M enthält die zu verschlüsselnden Daten. Der Parameter M ist ein Oktettstring mit beliebigem Inhalt. Die Länge von M MUSS so gewählt werden, dass für die Kommandonachricht die Vorgaben aus (N026.900) eingehalten werden und die Antwortnachricht nicht länger ist als *limitRspSecureMessaging*. **[<=]**

A_16655 - (N091.200) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *cipher* in der Antwortnachricht enthalten ist.**[<=]**

A_16656 - (N091.300) K_externeWelt {K_Karte}
Die Parameter *algID*, *oid*, PO_B und M MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N091.700)d spezifiziert.**[<=]**

A_16657 - (N091.400) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_b10 verwendet werden.

Tabelle 271: CosT_b10: PSO Encipher, Verschlüsseln mittels übergebenem ELC-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX…XX'	*plainDO*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.4.3 Use Case Verschlüsseln mittels gespeichertem RSA-Schlüssel

In dieser Variante wird ein in der Smartcard gespeicherter Schlüssel verwendet. Folgende Algorithmen sind zulässig: {rsaEncipherOaep}.

Hinweis CosH_ed7: Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher *Daten enthält.*

In dieser Variante enthält die APDU des PSO Encipher-Kommandos zwei Parameter:

A_18355 - (N091.420) K_externeWelt {K_Karte}, Option_Kryptobox
Der Parameter M enthält die zu verschlüsselnden Daten. Der Parameter M ist ein Oktettstring mit beliebigem Inhalt. Die maximal mögliche Länge von M ist abhängig von der mittels (N103.845) ausgewählten $algId$ und der Länge des Modulus n. **[<=]**

A_16658 - (N091.420)a K_externeWelt {K_Karte}, Option_Kryptobox
Falls $algId$ gleich rsaEncipherOaep ist, dann MUSS OctetLength(M) kleiner gleich OctetLength(n) – 66 sein.**[<=]**

(N091.420)b Diese Anforderung ist absichtlich leer.

A_16660 - (N091.422) K_externeWelt {K_Karte}, Option_Kryptobox
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.**[<=]**

A_16661 - (N091.424) K_externeWelt {K_Karte}, Option_Kryptobox
Es MUSS eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_c3c verwendet werden.

Tabelle 272: CosT_c3c: PSO Encipher, Verschlüsseln mittels gespeichertem RSA-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX…XX'	*M*
Le	'0000'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.4.4 Use Case Verschlüsseln mittels gespeichertem ELC-Schlüssel

In dieser Variante wird ein in der Smartcard gespeicherter Schlüssel verwendet. Folgende Algorithmen sind zulässig: {elcSharedSecretCalculation}.

Hinweis CosH_179: Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher *Daten enthält.*

In dieser Variante enthält die APDU des PSO Encipher-Kommandos zwei Parameter:

A_16662 - (N091.430) K_externeWelt {K_Karte}, Option_Kryptobox
Der Parameter *M* enthält die zu verschlüsselnden Daten. Der Parameter *M* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *M* MUSS so gewählt werden, dass für die Kommandonachricht die Vorgaben aus (N026.900) eingehalten werden und die Antwortnachricht nicht länger ist als *limitRspSecureMessaging*. **[<=]**

A_16663 - (N091.432) K_externeWelt {K_Karte}, Option_Kryptobox
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *cipher* in der Antwortnachricht enthalten ist.**[<=]**

A_16664 - (N091.434) K_externeWelt {K_Karte}, Option_Kryptobox
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_94d verwendet werden.

Tabelle 273: CosT_94d: PSO Encipher, Verschlüsseln mittels gespeichertem ELC-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX…XX'	*M*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.4.5 Use Case Verschlüsseln mittels symmetrischem Schlüssel

In dieser Variante wird ein in der Smartcard gespeicherter Schlüssel verwendet. Folgende Algorithmen sind zulässig: {aesSessionkey~~, desSessionkey (Option_DES)~~}.

Hinweis CosH_666: Gemäß den Festlegungen in (N091.650) und (N091.700) ist dieser Use Case nur dann ausführbar, wenn channelContext.keyReferenceList.dataEncipher *Daten enthält.*

In dieser Variante enthält die APDU des PSO Encipher-Kommandos zwei Parameter:

A_16665 - (N091.440) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *M* enthält die zu verschlüsselnden Daten. Der Parameter *M* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *M* MUSS so gewählt werden, dass für die Kommandonachricht die Vorgaben aus (N026.900) eingehalten werden und die Antwortnachricht nicht länger ist als *limitRspSecureMessaging*. **[<=]**

A_16666 - (N091.443) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *cipher* in der Antwortnachricht enthalten ist.**[<=]**

A_16667 - (N091.446) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_acb verwendet werden.

Tabelle 274: CosT_acb: PSO Encipher, Verschlüsseln mittels symmetrischem Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX…XX'	*M*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.4.6 Antwort der Karte auf Verschlüsseln von Daten

Tabelle 275: CosT_6f0: PSO Encipher Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*cipher*	Chiffrat
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	Erfolgreiche Verschlüsselungsoperation

Tabelle 276: CosT_6c0: PSO Encipher Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	EncipherError	Verschlüsselung fehlgeschlagen
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 88'	KeyNotFound	Schlüssel nicht gefunden

Hinweis CosH_37b: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16668 - (N091.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Encipher-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.4.7 Kommandoabarbeitung innerhalb der Karte

G2_N091.600.a - (N091.600)a K_COS
Das COS MUSS die PSO Encipher-Variante aus
- 14.8.4.1- Use Case Verschlüsseln von Daten mittels übergebenem RSA-Schlüssel und
- 14.8.4.2- Use Case Verschlüsseln von Daten mittels übergebenem ELC-Schlüssel
unterstützen.**[<=]**

G2_N091.600.b - (N091.600)b K_COS, Option_Kryptobox
Das COS MUSS die PSO Encipher-Variante aus
- 14.8.4.3- Use Case Verschlüsseln mittels gespeichertem RSA-Schlüssel und
- 14.8.4.4- Use Case Verschlüsseln mittels gespeichertem ELC-Schlüssel
unterstützen.**[<=]**

G2_N091.600.c - (N091.600)c K_COS, Option_Kryptobox oder Option_PACE_PCD
Das COS MUSS die PSO Encipher-Variante aus
- 14.8.4.5- Use Case Verschlüsseln mittels symmetrischem Schlüssel
unterstützen.**[<=]**

A_16669 - (N091.600)d K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Encipher-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16670 - (N091.650) K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *channelContext.keyReferenceList.dataEncipher* nicht leer ist, genau dann MUSS das Kommando auf folgende Art mit einem in der Smartcard gespeicherten Schlüssel arbeiten:**[<=]**

A_16671 - (N091.650)a K_COS
Es wird
affectedObject = SearchKey(
currentFolder,
keyReferenceList.dataEncipher.keyReference,
keyReferenceList.dataEncipher.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

A_16672 - (N091.650)a.1 K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

A_16673 - (N091.650)a.2 K_COS
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N091.650.b - (N091.650)b K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_17538 - (N091.650)c K_COS
Das Chiffrat *cipher* MUSS wie folgt berechnet werden:**[<=]**

G2_N091.650.c.1 - (N091.650)c.1 K_COS
Wenn *channelContext.keyReferenceList.dataEncipher.algID* den Wert aesSessionkey besitzt, dann gilt mit den Attributen *SSCmac* und *Kenc* aus dem Attribut *SessionkeyContext*:
0. Schritt 0: *SSCmac* = *SSCmac* + 1.
1. Schritt 1: *SSCenc* = OS2I(AES_ENC (*Kenc*, I2OS(*SSCmac*, 16))).
2. Schritt 2: *P* = PaddingIso(*M*, 16).
3. Schritt 3: *C* = AES_CBC_ENC(*Kenc*, *SSCenc*, *P*).
4. Schritt 4: *cipher* = '01' ||*C*. **[<=]**

(N091.650)c.2 ist absichtlich leer.

G2_N091.650.c.2 - (N091.650)c.2 K_COS, Option_DES
~~Falls *channelContext.keyReferenceList.dataEncipher.algID* den Wert desSessionkey besitzt, dann gilt mit den Attributen *SSCenc* und *Kenc* aus dem Attribut *SessionkeyContext*:~~
~~1. Schritt 1: *SSCenc* = *SSCenc* + 1.~~
~~2. Schritt 2: *P* = PaddingIso(*M*, 8).~~
~~3. Schritt 3: *C* = 3TDES_CBC_ENC(*Kenc*, *SSCenc*, *P*).~~
~~4. Schritt 4: *cipher* = '01' ||*C*.~~**[<=]**

G2_N091.650.c.3 - (N091.650)c.3 K_COS
Falls *channelContext.keyReferenceList.dataEncipher.algID* den Wert elcSharedSecretCalculation besitzt, dann MUSS gelten:
i. (PO_A, *C*, *T*) = ELC_ENC(*M*, *affectedObject.publicElcKey.P*, *affectedObject.domainParameter*).
Wenn diese Funktion mit dem Fehler "*ERROR*" terminiert, genau dann MUSS das Kommando mit dem Trailer EncipherError terminieren. Andernfalls wird ein DER-TLV codierter Oktettstring *cipher* wie folgt konstruiert:
Hinweis CosH_412: cipher *ist hier identisch zu (N090.300)c und (N094.400)c definiert.*
ii. Setze *oidDO* = '06 – L_{06} – *affectedObject.domainParameter*.OID'.
iii. Setze *keyDO* = '7F49 – L_{7F49} – (86 – L_{86} – PO_A)'.

iv. Setze *cipherDO* = '86 – L_{86} – (02 || *C*)'.
v. Setze *macDO* = '8E – L_{8E} – *T* '.
vi. Setze *cipher* = 'A6 – L_{A6} – (*oidDO* || *keyDO* || *cipherDO* || *macDO*)'.**[<=]**

G2_N091.650.c.4 - (N091.650)c.4 K_COS
Falls *channelContext.keyReferenceList.dataEncipher.algID* den Wert rsaEncipherOaep besitzt, dann gilt:
i. Schritt 1: *C* = RSAES_OAEP_ENCRYPT(*affectedObject.publicKey*, *M*)
ii. Schritt 2: *cipher* = '85' ||*C*. **[<=]**

(N091.650)c.5 Diese Anforderung ist absichtlich leer.

A_16674 - (N091.700) K_COS
Wenn *channelContext.keyReferenceList.dataEncipher* leer ist, genau dann MUSS das Kommando auf folgende Art mit einem Schlüssel in den Kommandodaten arbeiten: Das Chiffrat *cipher* MUSS wie folgt berechnet werden:**[<=]**

G2_N091.700.a - (N091.700)a K_COS
Suche in *plainDO* nach einem DO mit Tag = '80' und der Länge eins. Es gilt *algID_enc* = Wertfeld dieses DO.**[<=]**

(N091.700)b Diese Anforderung ist absichtlich leer.

A_16676 - (N091.700)c K_COS
Falls *algID_enc* den Wert rsaEncipherOaep besitzt, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N091.700.c.1 - (N091.700)c.1 K_COS
Schritt c.1: *plainDO* == 'A0 – L_{A0} – (*algDO* || *keyDO* || *mDO*).**[<=]**

G2_N091.700.c.2 - (N091.700)c.2 K_COS
Schritt c.2: *algDO* == '80 – 01 –*algID_enc* '.**[<=]**

G2_N091.700.c.3 - (N091.700)c.3 K_COS
Schritt c.3: *keyDO* == '7F49 – L_{7F49} – [(81 – L_{81} – *PuK.n*) || (82 – L_{82} – *PuK.e*)]'.**[<=]**

G2_N091.700.c.4 - (N091.700)c.4 K_COS
Schritt c.4: *mDO* == '80 – L_{80} – *M* '.**[<=]**

G2_N091.700.c.5 - (N091.700)c.5 K_COS
Schritt c.5: *cipher* = '00' || RSAES_OAEP_ENCRYPT(*PuK*, *M*).**[<=]**

A_16677 - (N091.700)d K_COS
Falls *algID_enc* den Wert elcSharedSecretCalculation besitzt, dann MÜSSEN folgende Schritte ausgeführt werden:
Hinweis CosH_737: cipher *ist hier identisch zu (N085.068)b.7, (N090.300)c und (N094.400)c definiert*. **[<=]**

G2_N091.700.d.1 - (N091.700)d.1 K_COS
Schritt d.1: *plainDO* == 'A0 – L_{A0} – (*algDO* || *oidDO* || *keyDO* || *mDO*).**[<=]**

G2_N091.700.d.2 - (N091.700)d.2 K_COS
Schritt d.2: *algDO* == '80 01 *algID_enc* '.**[<=]**

G2_N091.700.d.3 - (N091.700)d.3 K_COS
Schritt d.3: *oidDO* == '06 – L_{06} – *oid* '.**[<=]**

G2_N091.700.d.4 - (N091.700)d.4 K_COS
Schritt d.4: *keyDO* == '7F49 – L_{7F49} – (86 – L_{86} – PO_B)'.**[<=]**

G2_N091.700.d.5 - (N091.700)d.5 K_COS
Schritt d.5: mDO == '80 – L_{80} – M '.**[<=]**

G2_N091.700.d.6 - (N091.700)d.6 K_COS
Schritt d.6: Die oid aus der Kommandonachricht wird gemäß CosT_a91 in Domainparameter übersetzt, die im Folgenden mit dP bezeichnet werden.**[<=]**

G2_N091.700.d.7 - (N091.700)d.7 K_COS
Schritt d.7: (PO_A, C, T) = ELC_ENC(M, PO_B, dP).
Wenn diese Funktion mit dem Fehler "ERROR" terminiert, genau dann MUSS das Kommando mit dem Trailer EncipherError terminieren. Andernfalls wird ein DER-TLV codierter Oktettstring $cipher$ wie folgt konstruiert:**[<=]**

G2_N091.700.d.8 - (N091.700)d.8 K_COS
Schritt d.8: Setze $keyDO$ = '7F49 – L_{7F49} – (86 – L_{86} – PO_A)'.**[<=]**

G2_N091.700.d.9 - (N091.700)d.9 K_COS
Schritt d.9: Setze $cipherDO$ = '86 – L_{86} – (02 || C)'.**[<=]**

G2_N091.700.d.10 - (N091.700)d.10 K_COS
Schritt d.10: Setze $macDO$ = '8E – L_{8E} – T '.**[<=]**

G2_N091.700.d.11 - (N091.700)d.11 K_COS
Schritt d.11: Setze $cipher$ = 'A6 – L_{A6} – ($oidDO$ || $keyDO$ || $cipherDO$ || $macDO$).**[<=]**

(N091.800) Diese Anforderung ist absichtlich leer.

G2_N091.900 - (N091.900) K_COS
Als Datenfeld der Antwortnachricht MUSS $cipher$ verwendet werden.**[<=]**

G2_N092.000 - (N092.000) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16678 - (N092.100)a K_TST
Die Priorität der Trailer in CosT_6c0 MUSS herstellerspezifisch sein.**[<=]**

G2_N092.100.b - (N092.100)b K_COS
Jeder Trailer in CosT_6c0 MUSS eine höhere Priorität als NoError haben.**[<=]**

14.8.5 PSO Hash

A_16679 - (N092.200) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das PSO Hash-Kommando gemäß [ISO/IEC 7816-8]
a. unterstützt oder
b. ablehnt.**[<=]**

14.8.6 PSO Transcipher

Das Kommando PSO Transcipher kombiniert die Kommandos PSO Decipher und PSO Encipher, ohne dass die Klartextdaten die Karte verlassen. Es werden also Daten mittels eines privaten Schlüssels entschlüsselt und sofort wieder mittels eines öffentlichen Schlüssels verschlüsselt. Der private Schlüssel und der zu verwendende Algorithmus werden vor der Entschlüsselungsoperation ausgewählt. Dies geschieht vor dem Senden dieses PSO Transcipher-Kommandos durch ein MSE-Set-Kommando (siehe (N103.800)).

Die umzuschlüsselnden Daten und der öffentliche Schlüssel des Empfängers sind als Parameter in der Kommandonachricht enthalten.

14.8.6.1 Use Case Umschlüsseln von Daten mittels RSA-Schlüssel

In dieser Variante wird ein Chiffrat mittels eines privaten RSA-Schlüssels entschlüsselt und anschließend mit einem öffentlichen RSA-Schlüssels verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos vier Parameter:

A_16680 - (N092.300) K_externeWelt {K_Karte}
Der Parameter C_{in} enthält die umzuschlüsselnden Daten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.**[<=]**

A_16681 - (N092.400) K_externeWelt {K_Karte}
Der Parameter PuK enthält den öffentlichen Schlüssel des Empfängers gemäß 8.2.4.1. Der Parameter PuK ist ein Oktettstring, dessen Inhalt so gewählt werden MUSS, dass bei der Decodierung kein Fehler auftritt (siehe (N094.400)b.3).**[<=]**

A_16682 - (N092.500) K_externeWelt {K_Karte}
Der Parameter $algID_enc$ enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {rsaEncipherOaep} verwendet werden.**[<=]**

A_16683 - (N092.600) K_externeWelt {K_Karte}
Der Parameter $length$ bestimmt die Länge der erwarteten Antwortdaten. Der Wert von $length$ MUSS gleich WildCardExtended sein.**[<=]**

A_16684 - (N092.700) K_externeWelt {K_Karte}
Die Parameter C_{in}, PuK und $algID_enc$ MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.**[<=]**

A_16685 - (N092.800) K_externeWelt {K_Karte}
Es MUSS eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_f40 verwendet werden.

Tabelle 277: CosT_f40: PSO Transcipher, Umschlüsseln mittels RSA-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'B8'	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	'XX…XX'	$cipherIN$, DER codiertes Datenfeld
Le	'0000'	$length$, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.6.2 Use Case Umschlüsseln von Daten von RSA-Schlüssel nach ELC-Schlüssel

In dieser Variante wird ein Chiffrat mittels eines privaten RSA-Schlüssels entschlüsselt und anschließend mit einem öffentlichen ELC-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos fünf Parameter:

A_16686 - (N092.820) K_externeWelt {K_Karte}
Der Parameter C_{in} MUSS die umzuschlüsselnden Daten enthalten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.**[<=]**

A_16687 - (N092.821) K_externeWelt {K_Karte}
Der Parameter oid_{out} enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Verschlüsselung referenziert. Der Parameter oid_{out} MUSS aus der in CosT_a91 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.**[<=]**

A_16688 - (N092.822) K_externeWelt {K_Karte}
Der Parameter PO_B enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Empfängers. Der Parameter PO_B ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.500)a).**[<=]**

A_16689 - (N092.824) K_externeWelt {K_Karte}
Der Parameter $algID_enc$ enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {elcSharedSecretCalculation} verwendet werden.**[<=]**

A_16690 - (N092.826) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.**[<=]**

A_16691 - (N092.828) K_externeWelt {K_Karte}
Die Parameter C_{in}, oid_{out}, PO_B und $algID_enc$ MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.**[<=]**

A_16692 - (N092.830) K_externeWelt {K_Karte}
Es MUSS eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_632 verwendet werden.

Tabelle 278: CosT_632: PSO Transcipher, Umschlüsseln von RSA nach ELC

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'B8'	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	'XX…XX'	*cipherIN*
Le	'0000'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.6.3 Use Case Umschlüsseln von Daten mittels ELC

In dieser Variante wird ein Chiffrat mittels eines privaten ELC-Schlüssels entschlüsselt und anschließend mit einem öffentlichen ELC-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos acht Parameter:

A_16693 - (N092.900) K_externeWelt {K_Karte}
Der Parameter PO_A enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt wurde vom Sender der Nachricht gewählt. Der Parameter PO_A ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).**[<=]**

A_16694 - (N092.902) K_externeWelt {K_Karte}
Der Parameter oid_{in} enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Entschlüsselung referenziert. Der Parameter oid_{in} MUSS aus der in CosT_a91 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.**[<=]**

A_16695 - (N093.000) K_externeWelt {K_Karte}
Der Parameter C_{in} MUSS die umzuschlüsselnden Daten enthalten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.**[<=]**

A_16696 - (N093.100) K_externeWelt {K_Karte}
Der Parameter T_{in} enthält einen MAC, der die Integrität von C_{in} schützt. Der Parameter T_{in} ist ein Oktettstring, dessen Länge und Inhalt so gewählt werden SOLL, dass bei der MAC-Prüfung kein Fehler auftritt.**[<=]**

A_16697 - (N093.200) K_externeWelt {K_Karte}
Der Parameter PO_B enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Empfängers. Der Parameter PO_B ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).**[<=]**

A_16698 - (N093.202) K_externeWelt {K_Karte}
Der Parameter oid_{out} enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Verschlüsselung referenziert. Der Parameter oid_{out} MUSS aus der in CosT_a91 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.**[<=]**

A_16699 - (N093.300) K_externeWelt {K_Karte}
Der Parameter $algID_enc$ enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {elcSharedSecretCalculation} verwendet werden.**[<=]**

A_16700 - (N093.400) K_externeWelt {K_Karte}
Der Parameter $length$ bestimmt die Länge der erwarteten Antwortdaten. Der Wert von $length$ MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring $cipherOUT$ in der Antwortnachricht enthalten ist.**[<=]**

A_16701 - (N093.500) K_externeWelt {K_Karte}
Die Parameter PO_A, oid_{in}, C_{in}, T_{in}, PO_B, oid_{out} und $algID_enc$ MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.**[<=]**

A_16702 - (N093.600) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 oder eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09

geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_c68 verwendet werden.

Tabelle 279: CosT_c68: PSO Transcipher, Umschlüsseln mittels ELC-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'B8'	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	'XX…XX'	*cipherIN*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.6.4 Use Case Umschlüsseln von Daten von ELC-Schlüssel nach RSA-Schlüssel

In dieser Variante wird ein Chiffrat mittels eines privaten ELC-Schlüssels entschlüsselt und anschließend mit einem öffentlichen RSA-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos sieben Parameter:

A_16703 - (N093.620) K_externeWelt {K_Karte}
Der Parameter PO_A enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt wurde vom Sender der Nachricht gewählt. Der Parameter PO_A ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N004.800)a).**[<=]**

A_16704 - (N093.621) K_externeWelt {K_Karte}
Der Parameter oid_{in} enthält einen Objektidentifier, der die zu verwendende elliptische Kurve für die Entschlüsselung referenziert. Der Parameter oid_{in} MUSS aus der in CosT_a91 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.**[<=]**

A_16705 - (N093.622) K_externeWelt {K_Karte}
Der Parameter C_{in} MUSS die umzuschlüsselnden Daten enthalten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.**[<=]**

A_16706 - (N093.624) K_externeWelt {K_Karte}
Der Parameter T_{in} enthält einen MAC, der die Integrität von C schützt. Der Parameter T_{in} ist ein Oktettstring, dessen Länge und Inhalt so gewählt werden SOLL, dass bei der MAC-Prüfung kein Fehler auftritt.**[<=]**

A_16707 - (N093.626) K_externeWelt {K_Karte}
Der Parameter PuK enthält den öffentlichen Schlüssel des Empfängers gemäß 8.2.4.1. Der Parameter PuK ist ein Oktettstring, dessen Inhalt so gewählt werden MUSS, dass bei der Decodierung kein Fehler auftritt (siehe (N094.400)b.3).**[<=]**

A_16708 - (N093.628) K_externeWelt {K_Karte}
Der Parameter $algID_enc$ enthält den Algorithmus, der zur Verschlüsselung eingesetzt wird. Es MUSS ein Wert aus der Menge {rsaEncipherOaep} verwendet werden.**[<=]**

A_16709 - (N093.630) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.**[<=]**

A_16710 - (N093.632) K_externeWelt {K_Karte}
Die Parameter PO_A, oid_{in}, C_{in}, T_{in}, PuK und $algID_enc$ MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.**[<=]**

A_16711 - (N093.634) K_externeWelt {K_Karte}
Es MUSS eine Case 4E Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_8eb verwendet werden.

Tabelle 280: CosT_8eb: PSO Transcipher, Umschlüsseln von ELC nach RSA

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'86'	Beschreibung der Antwortdaten, hier Chiffrat
P2	'B8'	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	'XX…XX'	*cipherIN*
Le	'0000'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.8.6.5 Antwort der Karte auf Umschlüsseln von Daten

Tabelle 281: CosT_dba: PSO Transcipher Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*cipherOUT*	Chiffrat
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	Erfolgreiche Umschlüsselungsoperation

Tabelle 282: CosT_2dd: PSO Transcipher Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	KeyInvalid	Schlüsseldaten fehlen
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoKeyReference	Kein Schlüssel zur Entschlüsselung ausgewählt
'6A 80'	WrongCiphertext	Fehler beim Entschlüsseln des Chiffrats
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den geforderten Algorithmus nicht
'6A 84'	MessageTooLong	Klartext zu lang für Verschlüsselung
'6A 88'	KeyNotFound	Schlüssel für Entschlüsselung nicht gefunden

Hinweis CosH_619: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16712 - (N093.700) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Transcipher-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.6.6 Kommandoabarbeitung innerhalb der Karte

G2_N093.800.a - (N093.800)a K_COS
Das COS MUSS die PSO Transcipher-Variante aus
- 14.8.6.1- Use Case Umschlüsseln von Daten mittels RSA-Schlüssel,
- 14.8.6.2- Use Case Umschlüsseln von Daten von RSA-Schlüssel nach ELC-Schlüssel,
- 14.8.6.3- Use Case Umschlüsseln von Daten mittels ELC und
- 14.8.6.4- Use Case Umschlüsseln von Daten von ELC-Schlüssel nach RSA-Schlüssel

unterstützen.**[<=]**

A_16713 - (N093.800)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Transcipher-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N093.900.a - (N093.900)a K_COS
Wenn *channelContext.keyReferenceList.dataDecipher* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16714 - (N093.900)b K_COS
Wenn *channelContext.keyReferenceList.dataDecipher* nicht leer ist, dann wird *affectedObject* = SearchKey(
currentFolder,
keyReferenceList.dataDecipher.keyReference,
keyReferenceList.dataDecipher.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N093.900.b.1 - (N093.900)b.1 K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N093.900.b.2 - (N093.900)b.2 K_COS

Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N094.000 - (N094.000) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N094.010 - (N094.010) K_COS
Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.**[<=]**

A_16715 - (N094.100) K_COS
Die Kommandodaten MÜSSEN wie folgt aufgeteilt werden:
a. *cipherIN* == *cipher* || *plainDO*.
b. *cipher* MUSS ein DO mit Tag = 'A6' sein.
c. *plainDO* MUSS ein DO mit Tag = 'A0' sein.**[<=]**

A_16716 - (N094.200) K_COS
Der Klartext M MUSS in Abhängigkeit von *algID*
= *channelContext.keyReferenceList.dataDecipher.algID* wie folgt berechnet werden**[<=]**

(N094.200)a Diese Anforderung ist absichtlich leer.

G2_N094.200.b - (N094.200)b K_COS
Wenn *algID* den Wert rsaDecipherOaep besitzt, dann gilt:
1. *cipher* == 'A6 – L_{A6} – ($cipherDO_{in}$)'.
2. $cipherDO_{in}$ == '86 – L_{86} – (00 || C_{in})'.
3. M = RSAES_OAEP_DECRYPT(*affectedObject.privateRsaKey*, C_{in}).
Wenn diese Funktion mit dem Fehler "ERROR" terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.**[<=]**

G2_N094.200.c - (N094.200)c K_COS
Falls *algID* den Wert elcSharedSecretCalculation besitzt, dann gilt
(*Hinweis CosH_62d:* cipher *ist hier identisch zu (N090.300)c und (N091.700)d definiert*):
1. *cipher* == 'A6 – L_{A6} – ($oidDO_{in}$ || $keyDO_A$ || $cipherDO_{in}$ || $macDO_{in}$)'.
2. $oidDO_{in}$ == '06 – L_{06} – oid_{in} '.
3. $keyDO_A$ == '7F49 – L_{7F49} – (86 – L_{86} – PO_A)'.
4. $cipherDO_{in}$ == '86 – L_{86} – (02 || C_{in})'.
5. $macDO_{in}$ == '8E – L_{8E} – T_{in}'.
6. Falls oid_{in} verschieden ist zur OID, die gemäß (N008.600)d zu *affectedObject. privateElcKey. domainParameter* gehört, dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.
7. M = ELC_DEC(PO_A, *affectedObject.privateElcKey*, C_{in}, T_{in}).
Wenn diese Funktion mit dem Fehler "ERROR" terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren.**[<=]**

G2_N094.300 - (N094.300) K_COS
Suche in *plainDO* nach einem DO mit Tag = '80' dieses DO MUSS die Länge eins haben. Es gilt *algID_enc* = Wertfeld des DO mit Tag = '80'.**[<=]**

A_16717 - (N094.400) K_COS
Das Chiffrat *cipherOUT* MUSS wie folgt aus M berechnet werden:**[<=]**

(N094.400)a Diese Anforderung ist absichtlich leer.

G2_N094.400.b - (N094.400)b K_COS

Falls *algID_enc* den Wert rsaEncipherOaep besitzt, dann gilt:
1. *plainDO* == 'A0 – L_{A0} – (*algDO* || *keyDO*).
2. *algDO* == '80 – 01 –*algID_enc*'.
3. *keyDO* == '7F49 – L_{7F49} – [(81 – L_{81} – *PuK.n*) || (82 – L_{82} – *PuK.e*)]'.
4. *cipherOUT* = '00' || RSAES_OAEP_ENCRYPT(*PuK*, *M*).
Wenn diese Funktion mit dem Fehler "ERROR" terminiert, dann MUSS das Kommando mit dem Trailer MessageTooLong terminieren.**[<=]**

G2_N094.400.c - (N094.400)c K_COS
Falls *algID_enc* den Wert elcSharedSecretCalculation besitzt, dann gilt
(*Hinweis CosH_624:* cipherOUT *ist hier identisch zu (N085.068)b.7, (N090.300)c und* (N091.700)*d definiert*):
1. *plainDO* == 'A0 – L_{A0} – (*algDO* || $oidDO_{out}$ || $keyDO_B$).
2. *algDO* == '80 – 01 –*algID_enc*'.
3. $oidDO_{out}$ == '06 – L_{06} – oid_{out}'.
4. $keyDO_B$ == '7F49 – L_{7F49} – (86 – L_{86} – PO_B)'.
5. Die oid_{out} aus der Kommandonachricht wird gemäß CosT_a91 in Domainparameter übersetzt, die im Folgenden mit *dP* bezeichnet werden.
6. (PO_{out}, C_{out}, T_{out}) = ELC_ENC(*M*, PO_B, *dP*).
Wenn diese Funktion mit dem Fehler "ERROR" terminiert, genau dann MUSS das Kommando mit dem Trailer WrongCiphertext terminieren. Andernfalls wird ein DER-TLV codierter Oktettstring *cipherOUT* wie folgt konstruiert:
7. Setze $keyDO_{out}$ = '7F49 – L_{7F49} – (86 – L_{86} – PO_{out})'.
8. Setze $cipherDO_{out}$ = '86 – L_{86} – (02 || C_{out})'.
9. Setze $macDO_{out}$ = '8E – L_{8E} – T_{out}'.
10. Setze *cipherOUT* = 'A6 – L_{A6} – ($oidDO_{out}$ || $keyDO_{out}$ || $cipherDO_{out}$ || $macDO_{out}$).**[<=]**

G2_N094.500 - (N094.500) K_COS
Als Datenfeld der Antwortnachricht MUSS *cipherOUT* verwendet werden.**[<=]**

G2_N094.600 - (N094.600) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16718 - (N094.700)a K_TST
Die Priorität der Trailer in CosT_2dd MUSS herstellerspezifisch sein.**[<=]**

G2_N094.700.b - (N094.700)b K_COS
Jeder Trailer in CosT_2dd MUSS eine höhere Priorität als NoError haben.**[<=]**

14.8.7 PSO Verify Certificate

Das Kommando PSO Verify Certificate überprüft die Signatur eines Zertifikates mittels eines öffentlichen Schlüssels. Der öffentliche Schlüssel wird vor der Verifikationsoperation ausgewählt. Dies geschieht vor dem Senden dieses PSO Verify Certificate-Kommandos durch ein MSE-Set-Kommando (siehe (N103.300)). Das Zertifikat ist als Parameter in der Kommandonachricht enthalten. Falls die Signatur des Zertifikates als gültig betrachtet wird, dann werden gewisse Attribute des im Zertifikat enthaltenen Schlüsselobjektes zur späteren Verwendung gespeichert.

14.8.7.1 Use Case Importieren RSA-Schlüssels mittels Zertifikat, Option_RSA_CVC

Die folgenden Punkte sind absichtlich leer: (N094.800), (N094.810), (N094.820), (N094.900), (N095.000), (N095.100).

In dieser Variante enthält die APDU des PSO Verify Certificate-Kommandos zwei Parameter:

A_16719 - (N094.800) K_externeWelt {K_Karte}, Option_RSA_CVC
Der Parameter *certificateContent* MUSS Attributsinformationen für das Schlüsselobjekt enthalten.**[<=]**

G2_N094.810 - (N094.810) K_Anwendungsspezifikation {K_Karte}, Option_RSA_CVC
Die Anwendungsspezifikation einer Karte MUSS eine vollständige Liste aller CHA-Werte enthalten, die von dieser Karte zu unterstützen ist.**[<=]**

A_16720 - (N094.820) K_externeWelt {K_Karte}, Option_RSA_CVC
Der in *certificateContent* enthaltene Wert für CHA MUSS aus Element der Anwendungsspezifikation genannten Menge sein (siehe (N094.810)).**[<=]**

A_16721 - (N094.900) K_externeWelt {K_Karte}, Option_RSA_CVC
Der Parameter *signature* MUSS die von einer CA erstellte Signatur über die Attributsinformationen enthalten.**[<=]**

A_16722 - (N095.000) K_externeWelt {K_Karte}, Option_RSA_CVC
Die Parameter *certificateContent* und *signature* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N095.900)a spezifiziert.**[<=]**

A_16723 - (N095.100) K_externeWelt {K_Karte}, Option_RSA_CVC
Es MUSS eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_eb9 verwendet werden.

Tabelle 283, CosT_eb9: PSO Verify Certificate für RSA-Schlüssel

	Inhalt	**Beschreibung**
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Keine Antwortdaten
P2	'AE'	Kommandodaten mit Template, dessen Wertfelder zertifiziert sind
Data	'XX…XX'	*certificate*

[<=]

14.8.7.2 Use Case Importieren eines ELC-Schlüssels mittels Zertifikat

In dieser Variante enthält die APDU des PSO Verify Certificate-Kommandos zwei Parameter:

A_16724 - (N095.200) K_externeWelt {K_Karte}
Der Parameter *certificateContent* MUSS alle Attribute für das Schlüsselobjekt enthalten.**[<=]**

A_16725 - (N095.300) K_externeWelt {K_Karte}
Der Parameter *signature* MUSS die von einer CA erstellte Signatur über die Attribute enthalten.**[<=]**

A_16726 - (N095.400) K_externeWelt {K_Karte}
Die Parameter *certificateContent* und *signature* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N095.900)b.2 spezifiziert.**[<=]**

G2_N095.410 - (N095.410) K_COS
Der öffentliche Punkt *P* im Parameter *certificateContent* liegt gemäß [gemSpec_PKI] auf einer der Kurven brainpoolP256r1, brainpoolP384r1 oder brainpoolP512r1. Die Signature im Parameter *signature* ist gemäß [gemSpec_PKI] mit einem öffentlichen Schlüssel prüfbar, der auf einer der Kurven brainpoolP256r1, brainpoolP384r1 oder brainpoolP512r1 liegt. In einem CV-Zertifikat für elliptische Kurven ist eine Kurve für *certificateContent* zu kombinieren mit einer Kurve für *signature*. Für die zu unterstützenden Kombinationen durch das COS MÜSSEN die in CosT_952 genannten Schlüsselwörter gelten.

Tabelle 284: CosT_952: Kombination von Kurvenparametern in CV-Zertifikaten

certificateContent signature	brainpoolP256r1	brainpoolP384r1	brainpoolP512r1	andere
brainpoolP256r1	MUSS	MUSS	MUSS	KANN
brainpoolP384r1	MUSS	MUSS	MUSS	KANN
brainpoolP512r1	MUSS	MUSS	MUSS	KANN
andere	KANN	KANN	KANN	KANN

[<=]

A_16727 - (N095.500) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_8a7 verwendet werden.

Tabelle 285: CosT_8a7: PSO Verify Certificate für ELC-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Keine Antwortdaten
P2	'BE'	Kommandodaten mit zertifiziertem Template
Data	'XX…XX'	*certificate*

[<=]

14.8.7.3 Antwort den Importieren eines CV-Zertifikates

Tabelle 286, CosT_b8a: PSO Verify Certificate Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Zertifikatsprüfung

Tabelle 287: CosT_527: PSO Verify Certificate Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	KeyExpired	Gültigkeitszeitraum des a. Signaturprüfschlüssels ist abgelaufen b. Zertifikates ist abgelaufen
'69 85'	NoKeyReference	Kein Signaturprüfschlüssel ausgewählt
'6A 80'	VerificationError	Prüfung des Zertifikates fehlgeschlagen
'6A 88'	InconsistentKeyReference	Signaturprüfschlüssel hat eine andere Referenz als CAR des Zertifikates
'6A 88'	KeyNotFound	Signaturprüfschlüssel nicht gefunden

Hinweis CosH_7c5: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16728 - (N095.600) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Verify Certificate-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.7.4 Kommandoabarbeitung innerhalb der Karte

A_16729 - (N095.700)a.2 K_COS

Das COS MUSS die PSO Verify Certificate-Variante aus
~~- 14.8.7.1- Use Case Importieren RSA-Schlüssels mittels Zertifikat, Option_RSA_CVC und~~
- 14.8.7.2- Use Case Importieren eines ELC-Schlüssels mittels Zertifikat
unterstützen.**[<=]**

A_16730 - (N095.700)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Verify Certificate-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N095.800.a - (N095.800)a K_COS
Wenn *channelContext.keyReferenceList.verifyCertificate* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

G2_N095.800.b - (N095.800)b K_COS
Wenn *channelContext.keyReferenceList.verifyCertificate* nicht leer ist, dann wird *affectedObject* = SearchKey(
currentFolder,
keyReferenceList.verifyCertificate.keyReference,
verifyCertificate
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N095.810 - (N095.810) K_COS
Wenn *affectedObject* ein Attribut *expirationDate* besitzt und *affectedObject*.CHAT.*flagList* nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. Bits b0 b1 sind ungleich 11_2) und *affectedObject*. *expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.**[<=]**

G2_N095.820 - (N095.820) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

Je nach Typ des selektierten Signaturprüfschlüssel werden die Schlüsselinformationen wie folgt extrahiert:

(N095.900)a ist absichtlich leer.

A_16731 - (N095.900)a K_COS, Option_RSA_CVC
~~Falls *affectedObject.publicKey* vom Typ *publicRsaKey* ist, dann MÜSSEN folgende Schritte ausgeführt werden:~~**[<=]**

G2_N095.900.a.1 - (N095.900)a.1 K_COS, Option_RSA_CVC
~~*certificate* = '5F37 – L_{5F37} – *signature* || 5F38 – L_{5F38} – *certificateContent* '.~~**[<=]**

G2_N095.900.a.2 - (N095.900)a.2 K_COS, Option_RSA_CVC
~~(*out*, *M*) = RSA_ISO9796_2_DS1_VERIFY(*affectedObject.publicRsaKey*, *signature*, *certificateContent*).~~
~~Falls diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.~~**[<=]**

G2_N095.900.a.3 - (N095.900)a.3 K_COS, Option_RSA_CVC
~~Die Schlüsselattribute werden gemäß 7.1.2 aus der Nachricht *M* extrahiert und ein öffentliches Schlüsselobjekt *pukObj* gebildet.~~**[<=]**

A_16732 - (N095.900)a.4 K_COS, Option_RSA_CVC
~~Falls CPI gleich '21' ist, dann MUSS*pukObj* ein öffentliches Signaturprüfobjekt sein (siehe 8.6.4.1), dessen Attribute mit den Definitionen aus 7.1.2.1 wie folgt zu setzen sind:~~**[<=]**

G2_N095.900.a.4.i - (N095.900)a.4.i K_COS, Option_RSA_CVC
~~*pukObj.oid* = OID.~~**[<=]**

G2_N095.900.a.4.ii - (N095.900)a.4.ii K_COS, Option_RSA_CVC
~~*pukObj.publicKey.n* = Modulus.~~**[<=]**

G2_N095.900.a.4.iii - (N095.900)a.4.iii K_COS, Option_RSA_CVC
~~*pukObj.publicKey.e* = öffentlicherExponent.~~**[<=]**

G2_N095.900.a.4.iv - (N095.900)a.4.iv K_COS, Option_RSA_CVC
~~*pukObj.keyIdentifier* = CHR.~~**[<=]**

G2_N095.900.a.4.v - (N095.900)a.4.v K_COS, Option_RSA_CVC
~~*pukObj.lifeCycleStatus* = "Operational state (active)"~~**[<=]**

G2_N095.900.a.4.vi - (N095.900)a.4.vi K_COS, Option_RSA_CVC
~~*pukObj. accessRules = affectedObject. accessRulesPublicSignatureVerificationObject.*~~**[<=]**

G2_N095.900.a.4.vii - (N095.900)a.4.vii K_COS, Option_RSA_CVC
~~*pukObj.accessRulesPublicSignatureVerificationObject = affectedObject.accessRulesPublicSignatureVerificationObject.*~~**[<=]**

G2_N095.900.a.4.viii - (N095.900)a.4.viii K_COS, Option_RSA_CVC
~~*pukObj.accessRulesPublicAuthenticationObject = affectedObject.accessRulesPublicAuthenticationObject.*~~**[<=]**

A_16733 - (N095.900)a.5 K_COS, Option_RSA_CVC
~~Falls CPI gleich '22' ist, dann MUSS*pukObj* ein öffentliches Authentisierungsobjekt sein (siehe 8.6.4.2), dessen Attribute mit den Definitionen aus 7.1.2.2 wie folgt zu setzen sind:~~**[<=]**

G2_N095.900.a.5.i - (N095.900)a.5.i K_COS, Option_RSA_CVC
~~*pukObj.CHA* = CHA.~~**[<=]**

G2_N095.900.a.5.ii - (N095.900)a.5.ii K_COS, Option_RSA_CVC
~~*pukObj.oid* = OID.~~**[<=]**

G2_N095.900.a.5.iii - (N095.900)a.5.iii K_COS, Option_RSA_CVC
~~*pukObj.publicKey.n* = Modulus.~~**[<=]**

G2_N095.900.a.5.iv - (N095.900)a.5.iv K_COS, Option_RSA_CVC
~~*pukObj.publicKey.e* = öffentlicherExponent.~~**[<=]**

G2_N095.900.a.5.v - (N095.900)a.5.v K_COS, Option_RSA_CVC
~~*pukObj.keyIdentifier* = CHR.~~**[<=]**

G2_N095.900.a.5.vi - (N095.900)a.5.vi K_COS, Option_RSA_CVC
~~*pukObj.lifeCycleStatus* = "Operational state (active)"~~**[<=]**

G2_N095.900.a.5.vii - (N095.900)a.5.vii K_COS, Option_RSA_CVC
~~*pukObj. accessRules = affectedObject. accessRulesPublicAuthenticationObject.*~~**[<=]**

A_16751 - (N095.900)b K_COS
Wenn *affectedObject.publicKey* vom Typ *publicElcKey* ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N095.900.b.1.i - (N095.900)b.1.i K_COS
Das COS MUSS den Wert CPI = '70' = 112 unterstützen.**[<=]**

A_16752 - (N095.900)b.1.ii K_TST
Für die funktionale Eignung MUSS es zulässig sein, weitere CPI-Werte zu
A. unterstützen oder
B. abzulehnen.**[<=]**

G2_N095.900.b.2 - (N095.900)b.2 K_COS
certificate == '7F4E – L_{7F4E} – *certificateContent* || 5F37 – L_{5F37} – *signature* '.**[<=]**

G2_N095.900.b.3 - (N095.900)b.3 K_COS
Falls *affectedObject. publicElcKey. domainParameter. L* gleich
i. 32 ist, dann gilt: *hash* = SHA_256('7F4E – L_{7F4E} – *certificateContent* ').
ii. 48 ist, dann gilt: *hash* = SHA_384('7F4E – L_{7F4E} – *certificateContent* ').
iii. 64 ist, dann gilt: *hash* = SHA_512('7F4E – L_{7F4E} – *certificateContent* ').**[<=]**

G2_N095.900.b.4 - (N095.900)b.4 K_COS
signature wird wie folgt in R und S aufgeteilt:
signature == R || S, mit OctetLength(R) == OctetLength(S).**[<=]**

G2_N095.900.b.5 - (N095.900)b.5 K_COS
out = ELC_VER_SIG(*affectedObject.publicElcKey*, R, S, *hash*).
Wenn diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.**[<=]**

G2_N095.900.b.6 - (N095.900)b.6 K_COS
Wenn die OID im CHAT aus *certificateContent* verschieden ist von der aus *affectedObject. publickey. accessRights*,
dann MUSS das Kommando mit dem Trailer VerificationError terminieren.**[<=]**

A_16734 - (N095.900)b.7
Das COS MUSS das Wertfeld von CED als vorzeichenlose ganze Zahl interpretieren, gemäß *ced* = OS2I(Wertfeld_von_CED).**[<=]**

A_16735 - (N095.900)b.8
Das COS MUSS das Wertfeld von CXD als vorzeichenlose ganze Zahl interpretieren, gemäß *cxd* = OS2I(Wertfeld_von_CXD).**[<=]**

G2_N095.900.b.9 - (N095.900)b.9 K_COS
Wenn CED aus *certificateContent* größer ist als CXD aus *certificateContent*,
dann MUSS das Kommando mit dem Trailer VerificationError terminieren.**[<=]**

G2_N095.900.b.10 - (N095.900)b.10 K_COS
Das COS MUSS den Wert *effectiveFlagList* wie folgt bilden:
effectiveFlagList = *affectedObject*.CHAT.*flagList* AND (*flagList* aus *certificateContent*).**[<=]**

G2_N095.900.b.11.i - (N095.900)b.11.i K_COS
Wenn *effectiveFlagList* die Rolle Root-CA-Schlüssel anzeigt (d.h. Bits b0 b1 sind gleich 11_2),
dann DARF ein Import NICHT daran scheitern, dass CXD aus *certificateContent* größer ist als *affectedObject.expirationDate*.**[<=]**

G2_N095.900.b.11.ii - (N095.900)b.11.ii K_COS
Wenn *effectiveFlagList* nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. Bits b0 b1 sind ungleich 11_2) und CXD aus *certificateContent* größer ist als *affectedObject. expirationDate*,
dann MUSS das Kommando mit dem Trailer VerificationError terminieren.**[<=]**

G2_N095.900.b.12.i - (N095.900)b.12.i K_COS
Wenn *effectiveFlagList* die Rolle Root-CA-Schlüssel anzeigt (d.h. Bits b0 b1 sind gleich 11_2), dann DARF ein Import NICHT daran scheitern, dass CXD aus *certificateContent* kleiner als *pointInTime* ist (siehe (N019.900)j).**[<=]**

G2_N095.900.b.12.ii - (N095.900)b.12.ii K_COS
Wenn *effectiveFlagList* nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. Bits b0 b1 sind ungleich 11_2) und CXD aus *certificateContent* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.**[<=]**

G2_N095.900.b.13 - (N095.900)b.13 K_COS
Wenn CED aus *certificateContent* größer als *pointInTime* ist (siehe (N019.900)j), dann MUSS *pointInTime* mit Transaktionsschutz auf den Wert CED gesetzt werden.**[<=]**

A_16753 - (N095.900)b.14 K_COS
Die Schlüsselattribute MÜSSEN gemäß [gemSpec_PKI] aus *certificateContent* extrahiert und ein öffentliches Schlüsselobjekt *pukObj* gebildet werden.**[<=]**

G2_N095.900.b.15.i - (N095.900)b.15.i K_COS
Wenn das höchstwertige Bit von *effectiveFlagList* gesetzt ist,
dann MUSS *pukObj* ein öffentliches Signaturprüfobjekt sein (siehe 8.6.4.1).**[<=]**

G2_N095.900.b.15.ii - (N095.900)b.15.ii K_COS
Wenn das höchstwertige Bit von *effectiveFlagList* nicht gesetzt ist, dann MUSS *pukObj* ein öffentliches Authentisierungsobjekt sein (siehe 8.6.4.2).**[<=]**

A_16754 - (N095.900)b.16 K_COS
Die Attribute von *pukObj* MÜSSEN mit den Definitionen aus [gemSpec_PKI] wie folgt gesetzt werden:**[<=]**

G2_N095.900.b.16.i.A - (N095.900)b.16.i.A K_COS
Wenn das Wertfeld von DO'86' eine Länge von '41' = 65 hat,
dann gilt *PuK.domainParameter* = brainpoolP256r1.**[<=]**

G2_N095.900.b.16.i.B - (N095.900)b.16.i.B K_COS
Wenn das Wertfeld von DO'86' eine Länge von '61' = 97 hat,
dann gilt *PuK.domainParameter* = brainpoolP384r1.**[<=]**

G2_N095.900.b.16.i.C - (N095.900)b.16.i.C K_COS
Wenn das Wertfeld von DO'86' eine Länge von '81 '= 129 hat,
dann gilt *PuK.domainParameter* = brainpoolP512r1.**[<=]**

G2_N095.900.b.16.ii - (N095.900)b.16.ii K_COS
PuK. P = OS2P(Wertfeld von DO'86',*PuK.domainParameter*).**[<=]**

G2_N095.900.b.16.iii - (N095.900)b.16.iii K_COS
pukObj.keyIdentifier = CHR gemäß [gemSpec_PKI#6.7.2.4].**[<=]**

G2_N095.900.b.16.iv - (N095.900)b.16.iv K_COS
pukObj.lifeCycleStatus = "Operational state (active)".**[<=]**

A_16755 - (N095.900)b.16.v K_COS
pukObj.publicElcKey= *PuK*.**[<=]**

G2_N095.900.b.16.vi - (N095.900)b.16.vi K_COS
pukObj.oid = OID_{PuK}.
Hinweis CosH_00b: Anhand von pukObj.oid *ist erkennbar, ob* pukObj *ein Signaturprüfobjekt oder ein Authentisierungsobjekt ist.***[<=]**

A_16756 - (N095.900)b.16.vii K_COS
Wenn *pukObj* ein öffentliches Signaturprüfobjekt ist, dann MUSS gelten:**[<=]**

G2_N095.900.b.16.vii.A - (N095.900)b.16.vii.A K_COS
pukObj. accessRules = *affectedObject.accessRulesPublicSignatureVerificationObject*. **[<=]**

G2_N095.900.b.16.vii.B - (N095.900)b.16.vii.B K_COS
pukObj.accessRulesPublicSignatureVerificationObject = *affectedObject.accessRulesPublicSignatureVerificationObject*. **[<=]**

G2_N095.900.b.16.vii.C - (N095.900)b.16.vii.C K_COS
pukObj.accessRulesPublicAuthenticationObject = *affectedObject.accessRulesPublicAuthenticationObject*. **[<=]**

A_16757 - (N095.900)b.16.viii K_COS
Wenn *pukObj* ein öffentliches Authentisierungsobjekt ist, dann MUSS gelten:
pukObj. accessRules = *affectedObject. accessRulesPublicAuthenticationObject*.**[<=]**

G2_N095.900.b.16.ix - (N095.900)b.16.ix K_COS
pukObj.CHAT.OID_{flags} = *affectedObject*.CHAT.OID_{flags}. **[<=]**

G2_N095.900.b.16.x - (N095.900)b.16.x K_COS
pukObj.CHAT.*flagList* = *effectiveFlagList*.**[<=]**

G2_N095.900.b.16.xi - (N095.900)b.16.xi K_COS
pukObj.expirationDate = Wertfeld des Datenobjektes CXD.**[<=]**

G2_N095.900.c - (N095.900)c K_COS
Wenn *affectedObject.keyIdentifier* ungleich CAR aus dem importierten Zertifikat ist, genau dann MUSS das Kommando mit dem Trailer InconsistentKeyReference terminieren.**[<=]**

A_16747 - (N095.900)d K_COS
Das Objekt *pukObj* MUSS wie folgt gespeichert werden:**[<=]**

G2_N095.900.d.1 - (N095.900)d.1 K_COS
persistent = StoreInCache("Ordner dem *affectedObject* zugeordnet ist",*pukObj*).**[<=]**

A_16748 - (N096.000) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16749 - (N096.100) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N096.200 - (N096.200) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16750 - (N096.300) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_527 ist herstellerspezifisch.
b. Jeder Trailer in CosT_527 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.8.8 PSO Verify Cryptographic Checksum

Das Kommando PSO Verify Cryptographic Checksum überprüft mittels eines symmetrischen Schlüssels, ob eine gegebene kryptographische Checksumme zu gegebenen Daten passt. Der symmetrische Schlüssel wird im Rahmen einer gegenseitigen Authentisierung (siehe 15.4.1, 15.4.2, ~~15.4.3~~ und 15.4.4) ausgehandelt. Checksumme und geschützte Daten sind als Parameter in der Kommandonachricht enthalten.

14.8.8.1 Use Case Prüfen einer kryptographischen Checksumme

Diese Variante gilt für Algorithmen aus der folgenden Menge: {aesSessionkey~~, desSessionkey (Option_DES)~~}.

In dieser Variante enthält die APDU des PSO Verify Cryptographic Checksum-Kommandos zwei Parameter:

A_16758 - (N096.340) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *data* MUSS die geschützten Daten enthalten. Der Parameter *data* ist ein Oktettstring mit beliebigem Inhalt.**[<=]**

A_16759 - (N096.342) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Der Parameter *mac* enthält eine kryptographische Checksumme. Der Parameter *mac* ist ein Oktettstring mit beliebigem Inhalt, dessen Länge acht Oktette betragen MUSS.**[<=]**

Hinweis CosH_103: Falls mac *eine andere Länge als acht Oktett besitzt, dann schlägt gemäß ~~(N002.700)e und~~ (N002.810)h der Vergleich in (N002.900)b bzw. (N003.010)b oder (N003.020)b stets fehl.*

A_16760 - (N096.344) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Die Parameter *data* und *mac* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N096.366)a spezifiziert.**[<=]**

A_16761 - (N096.346) K_externeWelt {K_Karte}, Option_Kryptobox oder Option_PACE_PCD
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 oder eine Case 3E Kommando-APDU gemäß 11.7.3.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_674 verwendet werden.

Tabelle 288: CosT_674: PSO Verify Cryptographic Checksum, Prüfen MAC

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Beschreibung der Antwortdaten, hier keine Antwortdaten
P2	'A2'	Beschreibung der Kommandodaten, hier Input Template für MAC
Data	'XX…XX'	*inputTemplate*, DER codiertes Datenfeld

[<=]

14.8.8.2 Antwort der Karte auf Prüfen einer kryptographischen Checksumme

Tabelle 289: CosT_5e2: PSO Verify Cryptographic Checksum Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'90 00'	NoError	erfolgreiche Verifizierung eines MAC

Tabelle 290: CosT_804: PSO Verify Cryptographic Checksum Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	NoKeyReference	kein Schlüssel für MAC-Berechnung ausgewählt
'6A 80'	VerificationError	MAC-Prüfung fehlgeschlagen
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 88'	KeyNotFound	kein Schlüssel für MAC-Berechnung vorhanden

Hinweis CosH_6a5: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16762 - (N096.350) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Verify Cryptographic Checksum-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.8.3 Kommandoabarbeitung innerhalb der Karte

G2_N096.360.a - (N096.360)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Das COS MUSS die PSO Verify Cryptographic Checksum-Variante aus
- 14.8.8.1- Use Case Prüfen einer kryptographischen Checksumme

unterstützen.**[<=]**

A_16763 - (N096.360)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Verify Cryptographic Checksum-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N096.362.a - (N096.362)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *channelContext.keyReferenceList.macCalculation* leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.**[<=]**

A_16764 - (N096.362)b K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *channelContext.keyReferenceList.macCalculation* nicht leer ist, dann wird *affectedObject* = SearchKey(
currentFolder,
keyReferenceList.macCalculation.keyReference,
keyReferenceList.macCalculation.algID
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist.**[<=]**

G2_N096.362.b.1 - (N096.362)b.1 K_COS, Option_Kryptobox oder Option_PACE_PCD

Wenn die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N096.362.b.2 - (N096.362).b.2 K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn die Schlüsselsuche den Fehler notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N096.364 - (N096.364) K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_16765 - (N096.366)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Es gilt: *inputTemplate* == '80 – L_{80} – *data* || 8E – L_{8E} – *mac* '.**[<=]**

Mit den Attributen *SSCmac* und *Kmac* aus dem Attribut *SessionkeyContext* sowie algID = *keyReferenceList.macCalculation.algID* gilt:

G2_N096.368.a - (N096.368)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Falls *algID* den Wert aesSessionkey besitzt, gilt:
1. *SSCmac* = *SSCmac* + 1.
2. *result* = VerifyCMAC_IsoPadding(*Kmac*, *mac*, I2OS(*SSCmac*, 16) || *data*).**[<=]**

(N096.368)b ist absichtlich leer.

G2_N096.368.b - (N096.368)b K_COS, (Option_Kryptobox oder Option_PACE_PCD) und Option_DES
~~Falls *algID* den Wert desSessionkey besitzt, gilt:~~
~~1. *SSCmac* = *SSCmac* + 1.~~
~~2. *result* = VERIFY_Retail_MAC(*Kmac*, *mac*, I2OS(*SSCmac*, 8) || *data*).~~**[<=]**

G2_N096.370.a - (N096.370.)a K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *result* den Wert INVALID besitzt, dann
1. MUSS als Trailer VerificationError verwendet werden und
2. die Sessionkeys MÜSSEN mittels clearSessionkeys( ) gelöscht werden.**[<=]**

G2_N096.370.b - (N096.370)b K_COS, Option_Kryptobox oder Option_PACE_PCD
Wenn *result* den Wert VALID besitzt, genau dann MUSS als Trailer NoError verwendet werden.**[<=]**

A_16766 - (N096.372)a K_TST
Die Priorität der Trailer in CosT_804 MUSS herstellerspezifisch sein.**[<=]**

G2_N096.372.b - (N096.372.b K_COS, Option_Kryptobox oder Option_PACE_PCD
Jeder Trailer in CosT_804 MUSS eine höhere Priorität als NoError haben.**[<=]**

14.8.9 PSO Verify Digital Signature

Das Kommando PSO Verify Digital Signature überprüft eine Signatur mittels eines öffentlichen Schlüssels. Sowohl die Signatur, als auch der öffentliche Schlüssel und der zur Prüfung zu verwendende Algorithmus sind als Parameter in der Kommandonachricht enthalten.

14.8.9.1 Use Case Prüfen einer ELC-Signatur

Diese Variante prüft Signaturen, die mittels signECDSA erstellt wurden. Als Domainparameter sind alle Kurven zulässig, die vom COS unterstützt werden (siehe (N002.500)).

In dieser Variante enthält die APDU des PSO Verify Digital Signature vier Parameter:

A_16767 - (N096.380) K_externeWelt {K_Karte}
Der Parameter *oid* enthält einen Objektidentifier, der die zu verwendende elliptische Kurve referenziert. Der Parameter *oid* MUSS aus der in CosT_a91 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.**[<=]**

A_16768 - (N096.381) K_externeWelt {K_Karte}
Der Parameter PO_B enthält einen Punkt auf einer elliptischen Kurve. Dieser Punkt repräsentiert den öffentlichen Schlüssel des Signierenden. Der Parameter PO_B ist ein Oktettstring, dessen Inhalt so gewählt werden SOLL, dass bei der Decodierung kein Fehler auftritt (siehe (N096.396)a Funktion O2P(...) und (N000.300)~~(N004.500)a~~).**[<=]**

A_16769 - (N096.382) K_externeWelt {K_Karte}
Der Parameter *hash* enthält den im Rahmen der Signaturerstellung erzeugten Hashwert. Der Parameter *hash* ist ein Oktettstring mit beliebigem Inhalt. Für den Zusammenhang zwischen *oid* und der Länge von *hash* MUSS gelten:
a. ansix9p256r1 => OctetLength(*hash*) gleich 32.
b. ansix9p384r1 => OctetLength(*hash*) gleich 48.
c. brainpoolP256r1 => OctetLength(*hash*) gleich 32.
d. brainpoolP384r1 => OctetLength(*hash*) gleich 48.
e. brainpoolP512r1 => OctetLength(*hash*) gleich 64.**[<=]**

A_16770 - (N096.383) K_externeWelt {K_Karte}
Der Parameter *signature* enthält die zu prüfende Signatur. Der Parameter *signature* ist ein Oktettstring mit beliebigem Inhalt. Für den Zusammenhang zwischen *oid* und der Länge von *signature* MUSS gelten:
a. ansix9p256r1 => OctetLength(*signature*) gleich 64.
b. ansix9p384r1 => OctetLength(*signature*) gleich 96.
c. brainpoolP256r1 => OctetLength(*signature*) gleich 64.
d. brainpoolP384r1 => OctetLength(*signature*) gleich 96.
e. brainpoolP512r1 => OctetLength(*signature*) gleich 128.**[<=]**

A_16771 - (N096.386) K_externeWelt {K_Karte}
Die Parameter *oid*, PO_B, *hash* und *signature* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N096.394) spezifiziert.**[<=]**

A_16772 - (N096.388) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_f4d verwendet werden.

Tabelle 291: CosT_f4d: PSO Verify Digital Signature mittels übergebenem ELC-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'2A'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	Keine Antwortdaten
P2	'A8'	Beschreibung der Kommandodaten, Template einer digitalen Signatur
Data	'XX…XX'	*signatureTemplate*, DER codiertes Datenfeld

[<=]

14.8.9.2 Antwort der Karte auf Prüfen einer digitalen Signatur

Tabelle 292: CosT_c29: PSO Verify Digital Signature Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'90 00'	NoError	erfolgreiche Verifizierung einer Signatur

Tabelle 293: CosT_3f2: PSO Verify Digital Signature Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'6A 80'	VerificationError	Signaturprüfung fehlgeschlagen

Hinweis CosH_67f: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16773 - (N096.390) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das PSO Verify Digital Signature-Kommando zusätzliche Trailer verwendet.**[<=]**

14.8.9.3 Kommandoabarbeitung innerhalb der Karte

G2_N096.392.a - (N096.392)a K_COS
Das COS MUSS die PSO Verify Digital Signature-Variante aus
- 14.8.9.1- Use Case Prüfen einer ELC-Signatur

unterstützen.**[<=]**

A_16774 - (N096.392)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere PSO Verify Digital Signature-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16775 - (N096.394) K_COS
Mit den Festlegungen aus [ISO/IEC 7816-8#Table 6] MUSS gelten:
a. *signatureTemplate* == '(06 – L_{06} – *oid*) || (90 – L_{90} – *hash*) || (9C – L_{9C} – *publicKey*) || (9E – L_{9E} – *signature*)'.
b. *publicKey* == '7F49 – L_{7F49} – (86 – L_{86} – PO_B)'.
c. *signature* wird wie folgt auf R und S aufgeteilt:
   1. *signature* == R || S.

2. OctetLength(R) == OctetLength(S).

d. Die oid wird gemäß CosT_a91 in Domainparameter übersetzt, die im Folgenden mit dP bezeichnet werden.**[<=]**

G2_N096.396.a - (N096.396)a K_COS
Falls ELC_VER_SIG(OS2P(PO_B, dP), R, S, $hash$) den Wert False besitzt, genau dann MUSS als Trailer VerificationError verwendet werden.**[<=]**

G2_N096.396.b - (N096.396)b K_COS
Falls ELC_VER_SIG(OS2P(PO_B, dP), R, S, $hash$) den Wert True besitzt, genau dann MUSS als Trailer NoError verwendet werden.**[<=]**

A_16776 - (N096.398)a K_TST
Die Priorität der Trailer in CosT_3f2 MUSS herstellerspezifisch sein.**[<=]**

G2_N096.398.b - (N096.398)b K_COS
Jeder Trailer in CosT_3f2 MUSS eine höhere Priorität als NoError haben.**[<=]**

14.9 Verschiedenes

14.9.1 ENVELOPE

A_16777 - (N096.400) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das ENVELOPE-Kommando gemäß [ISO/IEC 7816-4]
a. unterstützt oder
b. ablehnt.**[<=]**

14.9.2 FINGERPRINT

Das Kommando FINGERPRINT dient der Überprüfung der Integrität und Authentizität des COS. Dazu wird im Kommando ein Präfix übergeben. Über das Präfix und das COS wird ein Fingerprint berechnet. Stimmen berechneter Fingerprint und der Fingerprint eines authentischen COS überein, so ist die Authentizität des COS nachgewiesen.

Hinweis CosH_2f6: Typischerweise werden für Tests im Rahmen einer funktionalen Zulassung und im Rahmen einer Sicherheitsevaluierung verschiedene Images erzeugt. Zudem werden typischerweise im Rahmen einer Fehlerbeseitigung Patches entwickelt. Mit Hilfe dieses Kommandos lässt sich der Nachweis führen, dass in sämtlichen Images, die für ein Zulassungsverfahren relevant sind, derselbe Softwarestand (im Allgemeinen ROM Code plus Patches) zu Grunde liegt.

14.9.2.1 Use Case Berechnen eines COS-Fingerprints

In dieser Variante enthält die APDU des FINGERPRINT-Kommandos zwei Parameter:

A_16778 - (N096.450) K_externeWelt {K_Karte}
Der Parameter *prefix* enthält den Präfix. Der Parameter *prefix* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *prefix* MUSS 128 Oktett betragen.**[<=]**

A_16779 - (N096.452) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS WildCardShort sein.**[<=]**

A_16780 - (N096.454) K_externeWelt {K_Karte}
Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus CosT_b85 verwendet werden.

Tabelle 294: CosT_b85: FINGERPRINT über das COS

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'FA'	Instruction Byte
P1	'00'	-
P2	'00'	–
Data	'XX…XX'	*prefix*
Le	'00'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.2.2 Antwort der Karte auf Berechnen eines COS-Fingerprints

Tabelle 295: CosT_bee: FINGERPRINT Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'xx…xx'	*fingerprint*	Fingerprint des COS zum gegebenen Präfix
'90 00'	NoError	Erfolgreiche Fingerprintberechnung

Tabelle 296: CosT_105: FINGERPRINT Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis CosH_252: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16781 - (N096.469) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das FINGERPRINT-Kommando zusätzliche Trailer verwendet.**[<=]**

14.9.2.3 Kommandoabarbeitung innerhalb der Karte

G2_N096.470.a - (N096.470)a K_COS
Das COS MUSS die FINGERPRINT-Variante aus
- 14.9.2.1- Use Case Berechnen eines COS-Fingerprints
unterstützen.**[<=]**

A_16782 - (N096.470)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere FINGERPRINT-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N096.472 - (N096.472) K_COS
Als *affectedObject* MUSS *currentFolder* verwendet werden.**[<=]**

G2_N096.474 - (N096.474) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N096.476 - (N096.476) K_COS
Berechnung des Repräsentanten M des COS:
a. Das COS MUSS einen Oktettstring M konstruieren, der sämtliche Bestandteile des COS beinhaltet.
b. Die Funktion, die das COS auf M abbildet ist herstellerspezifisch und
   1. MUSS zeitinvariant sein,
   2. MUSS reproduzierbar sein und
   3. DARF NICHT abhängig sein von Veränderungen am Objektsystem, die mit Kommandos möglich sind, die im Rahmen dieser Spezifikation möglich und gemäß Objektsystemspezifikation erlaubt sind.**[<=]**

G2_N096.478 - (N096.478) K_COS
Basierend auf dem Parameter *prefix* aus den Kommandodaten und M MUSS das Datenfeld *fingerprint* der Antwortnachricht mit einem der folgenden Verfahren berechnet werden:
a. *fingerprint* = SHA_256(*prefix* || M) gesetzt werden.
b. *fingerprint* = SHA_384(*prefix* || M) gesetzt werden.
c. *fingerprint* = SHA_512(*prefix* || M) gesetzt werden.
d. *fingerprint* = CalculateCMAC_IsoPadding(*key*, *prefix* || M) gesetzt werden, wobei *key* ein herstellerspezifischer Schlüssel ist.**[<=]**

G2_N096.480 - (N096.480) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

G2_N096.482 - (N096.482) K_COS
Das Datenfeld der Antwortnachricht MUSS *fingerprint* sein.**[<=]**

14.9.3 GENERATE ASYMMETRIC KEY PAIR

Das Kommando GENERATE ASYMMETRIC KEY PAIR (GAKP) dient dem Erzeugen von asymmetrischen Schlüsselpaaren und dem Auslesen eines dabei erzeugten öffentlichen Schlüssels. Es ist möglich das betroffen Schlüsselobjekt zuvor auszuwählen. Dies geschieht durch ein MSE-Set-Kommando (siehe (N102.900)). Zusätzlich ist es möglich, den zu generierenden Schlüssel im Kommando zu referenzieren.

14.9.3.1 Use Case Generieren, ohne Überschreiben, ohne Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MSE-Set-Kommando ausgewählt und mit Schlüsseldaten befüllt, falls diese fehlen. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier einen Parameter:

A_16783 - (N096.500) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '84' gewählt werden.**[<=]**

A_16784 - (N096.600) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_d52 verwendet werden.

Tabelle 297: CosT_d52: GAKP, ohne Überschreiben, ohne Referenz, ohne Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'84'	*operationMode* = Schlüsselgenerierung, falls kein Schlüssel vorhanden
P2	'00'	betroffenes Objekt via *channelContext.keyReferenceList*

[<=]

14.9.3.2 Use Case Generieren, ohne Überschreiben, mit Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt und mit Schlüsseldaten befüllt, falls diese fehlen. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

A_16785 - (N096.640) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '84' gewählt werden.**[<=]**

A_16786 - (N096.642) K_externeWelt {K_Karte}
Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.**[<=]**

A_16787 - (N096.644) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_39e verwendet werden.

Tabelle 298: CosT_39e: GAKP, ohne Überschreiben, mit Schlüsselreferenz, ohne Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'84'	*operationMode* = Schlüsselgenerierung, falls kein Schlüssel vorhanden
P2	'XX'	*keyReference* bestimmt betroffenes Schlüsselobjekt

[<=]

14.9.3.3 Use Case Generieren, ggf. Überschreiben, ohne Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MSE-Set-Kommando ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Die APDU des GAKP-Kommandos enthält hier einen Parameter:

A_16788 - (N096.650) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C4' gewählt werden.**[<=]**

A_16789 - (N096.652) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_8ee verwendet werden.

Tabelle 299: CosT_8ee: GAKP, ggf. Überschreiben, ohne Referenz, ohne Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'C4'	*operationMode* = Schlüsselgenerierung, ggf. Überschreiben
P2	'00'	betroffenes Objekt via *channelContext.keyReferenceList*

[<=]

14.9.3.4 Use Case Generieren, ggf. Überschreiben, mit Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

A_16790 - (N096.660) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C4' gewählt werden.**[<=]**

A_16791 - (N096.662) K_externeWelt {K_Karte}
Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.**[<=]**

A_16792 - (N096.664) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_e3e verwendet werden.

Tabelle 300: CosT_e3e: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, ohne Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'C4'	*operationMode* = Schlüsselgenerierung, ggf. Überschreiben
P2	'XX'	*keyReference* bestimmt betroffenes Schlüsselobjekt

[<=]

14.9.3.5 Use Case Auslesen vorhandener Schlüssel, ohne Referenz

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MSE-Set-Kommando ausgewählt. Die dort vorhandenen Schlüsseldaten werden ausgelesen. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

A_16793 - (N096.700) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.**[<=]**

A_16794 - (N096.800) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* (siehe (N098.200){a, b}) in der Antwortnachricht enthalten ist.**[<=]**

A_16795 - (N096.900) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_36f verwendet werden.

Tabelle 301: CosT_36f: GAKP, Auslesen vorhandener Schlüssel ohne Schlüsselreferenz

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	*operationMode* = Auslesen eines öffentlichen Schlüssels
P2	'00'	betroffenes Objekt via *channelContext.keyReferenceList*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.3.6 Use Case Auslesen vorhandener Schlüssel, mit Referenz

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt. Die dort vorhandenen Schlüsseldaten werden ausgelesen. Die APDU des GAKP-Kommandos enthält hier drei Parameter:

A_16796 - (N096.940) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.**[<=]**

A_16797 - (N096.942) K_externeWelt {K_Karte}
Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.**[<=]**

A_16798 - (N096.944) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* (siehe (N098.200){a, b}) in der Antwortnachricht enthalten ist.**[<=]**

A_16799 - (N096.946) K_externeWelt {K_Karte}

Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando–APDU MÜSSEN die Angaben aus CosT_f6d verwendet werden.

Tabelle 302: CosT_f6d: GAKP, Auslesen vorhandener Schlüssel mit Schlüsselreferenz

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	*operationMode* = Auslesen eines öffentlichen Schlüssels
P2	'XX'	*keyReference* bestimmt betroffenes Schlüsselobjekt
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.3.7 Use Case Generieren, ohne Überschreiben, ohne Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MSE-Set-Kommando ausgewählt, mit Schlüsseldaten befüllt und die der erzeugte öffentliche Schlüssel wird exportiert. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

A_16800 - (N097.000) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '80' gewählt werden.**[<=]**

A_16801 - (N097.100) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* (siehe (N098.200){a, b}) in der Antwortnachricht enthalten ist.**[<=]**

A_16802 - (N097.200) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_fb7 verwendet werden.

Tabelle 303: CosT_fb7: GAKP, ohne Überschreiben, ohne Schlüsselreferenz, mit Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'80'	*operationMode* = Schlüsselgenerierung falls nicht vorhanden, Ausgabe
P2	'00'	betroffenes Objekt via *channelContext.keyReferenceList*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.3.8 Use Case Generieren, ohne Überschreiben, mit Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt, mit Schlüsseldaten befüllt und der öffentliche Teil des erzeugten Schlüsselpaares wird exportiert. Falls bereits Schlüsseldaten vorhanden sind, bleiben diese erhalten. Die APDU des GAKP-Kommandos enthält hier drei Parameter:

In dieser Variante enthält die APDU des GAKP-Kommandos drei Parameter:

A_16803 - (N097.240) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '80' gewählt werden.**[<=]**

A_16804 - (N097.242) K_externeWelt {K_Karte}
Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.**[<=]**

A_16805 - (N097.244) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* (siehe (N098.200){a, b}) in der Antwortnachricht enthalten ist.**[<=]**

A_16806 - (N097.246) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_c33 verwendet werden.

Tabelle 304: CosT_c33: GAKP, ohne Überschreiben, mit Schlüsselreferenz, mit Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'80'	*operationMode* = Schlüsselgenerierung falls nicht vorhanden, Ausgabe
P2	'XX'	*keyReference* bestimmt betroffenes Schlüsselobjekt
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.3.9 Use Case Generieren, ggf. Überschreiben, ohne Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MSE-Set-Kommando ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Der erzeugte öffentliche Schlüssel wird exportiert. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

A_16807 - (N097.250) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C0' gewählt werden.**[<=]**

A_16808 - (N097.252) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden,

dass der komplette Oktettstring *publicKeyDO* (siehe (N098.200){a, b}) in der Antwortnachricht enthalten ist.**[<=]**

A_16809 - (N097.254) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_5d8 verwendet werden.

Tabelle 305: CosT_5d8: GAKP, ggf. Überschreiben, ohne Schlüsselreferenz, mit Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'C0'	*operationMode* = Schlüsselgenerierung, ggf. Überschreiben, Ausgabe
P2	'00'	betroffenes Objekt via *channelContext.keyReferenceList*
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.3.10 Use Case Generieren, ggf. Überschreiben, mit Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch den Parameter P2 ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Der erzeugte öffentliche Schlüssel wird exportiert. Die APDU des GAKP-Kommandos enthält hier drei Parameter:

A_16810 - (N097.260) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C0' gewählt werden.**[<=]**

A_16811 - (N097.262) K_externeWelt {K_Karte}
Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.**[<=]**

A_16812 - (N097.264) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS so aus der Menge {WildCardShort, WildCardExtended} gewählt werden, dass der komplette Oktettstring *publicKeyDO* (siehe (N098.200){a, b}) in der Antwortnachricht enthalten ist.**[<=]**

A_16813 - (N097.266) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 oder eine Case 2E Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_cd2 verwendet werden.

Tabelle 306: CosT_cd2: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, mit Ausgabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'C0'	*operationMode* = Schlüsselgenerierung, ggf. Überschreiben, Ausgabe
P2	'XX'	*keyReference* bestimmt betroffenes Schlüsselobjekt
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.3.11 Zusammenfassung der GENERATE ASYMMETRIC KEY PAIR-Kommando-Varianten

Wegen der Vielzahl an Varianten für dieses Kommando werden hier alle auf einen Blick dargestellt. Es sei darauf hingewiesen, dass nicht alle Kombinationen der folgenden Tabelle in den vorangegangenen Kapiteln enthalten sind. Welche der möglichen Kombinationen zu unterstützen sind wird in (N097.400) festgelegt.

Tabelle 307: CosT_246: GENERATE ASYMMETRIC KEY PAIR, Kommandoparameter im Überblick

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'46'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81' '80' '84' 'C0' 'C4'	keine Schlüsselgenerierung, nur Ausgabe Schlüsselgenerierung, falls kein Schlüssel vorhanden und Ausgabe Schlüsselgenerierung, falls kein Schlüssel vorhanden, keine Ausgabe Schlüsselgenerierung, ggf. Überschreiben und Ausgabe Schlüsselgenerierung, ggf. Überschreiben, keine Ausgabe
P2	'00' sonst	betroffenes Objekt via *channelContext.keyReferenceList* *keyReference* bestimmt betroffenes Schlüsselobjekt
Le	*length*	Bit b3 von P1 == 0: Anzahl der erwarteten Oktette in den Antwortdaten Bit b3 von P1 == 1: abwesend

14.9.3.12 Antwort der Karte auf Generieren oder Auslesen eines asym. Schlüssels

Tabelle 308: CosT_a45: GENERATE ASYMMETRIC KEY PAIR Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*publicKeyDO*	Abwesend oder öffentlicher Schlüssel
Trailer	**Inhalt**	**Beschreibung**
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Operation

Tabelle 309: CosT_6d3: GENERATE ASYMMETRIC KEY PAIR Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'64 00'	KeyInvalid	Auszulesende Schlüsseldaten fehlen
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 85'	KeyAlreadyPresent	Schlüsseldaten bereits gesetzt, Generierung unmöglich
'6A 88'	KeyNotFound	Referenziertes Schlüsselobjekt wurde nicht gefunden

Hinweis CosH_c72: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16814 - (N097.300) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das GENERATE ASYMMETRIC KEY PAIR-Kommando zusätzliche Trailer verwendet.**[<=]**

14.9.3.13 Kommandoabarbeitung innerhalb der Karte

G2_N097.400.a - (N0097.400)a.1 K_COS
Das COS MUSS für private ELC-Schlüssel die GENERATE ASYMMETRIC KEY PAIR-Varianten aus
- 14.9.3.1- Use Case Generieren, ohne Überschreiben, ohne Referenz, ohne Ausgabe,
- 14.9.3.2- Use Case Generieren, ohne Überschreiben, mit Referenz, ohne Ausgabe,
- 14.9.3.3- Use Case Generieren, ggf. Überschreiben, ohne Referenz, ohne Ausgabe,
- 14.9.3.4- Use Case Generieren, ggf. Überschreiben, mit Referenz, ohne Ausgabe,
- 14.9.3.5- Use Case Auslesen vorhandener Schlüssel, ohne Referenz,
- 14.9.3.6- Use Case Auslesen vorhandener Schlüssel, mit Referenz,
- 14.9.3.7- Use Case Generieren, ohne Überschreiben, ohne Referenz, mit Ausgabe,
- 14.9.3.8- Use Case Generieren, ohne Überschreiben, mit Referenz, mit Ausgabe,
- 14.9.3.9- Use Case Generieren, ggf. Überschreiben, ohne Referenz, mit Ausgabe und
- 14.9.3.10- Use Case Generieren, ggf. Überschreiben, mit Referenz, mit Ausgabe

unterstützen.**[<=]**

A_16816 - (N097.400)a.2 K_COS
Das COS MUSS für private RSA-Schlüssel die GENERATE ASYMMETRIC KEY PAIR-Varianten aus
- 14.9.3.5- Use Case Auslesen vorhandener Schlüssel, ohne Referenz und
- 14.9.3.6- Use Case Auslesen vorhandener Schlüssel, mit Referenz,

unterstützen.**[<=]**

A_16817 - (N097.400)a.3 K_COS, Option_RSA_KeyGeneration
Das COS MUSS für private RSA-Schlüssel die GENERATE ASYMMETRIC KEY PAIR-Varianten aus
- 14.9.3.1- Use Case Generieren, ohne Überschreiben, ohne Referenz, ohne Ausgabe,
- 14.9.3.2- Use Case Generieren, ohne Überschreiben, mit Referenz, ohne Ausgabe,
- 14.9.3.3- Use Case Generieren, ggf. Überschreiben, ohne Referenz, ohne Ausgabe,
- 14.9.3.4- Use Case Generieren, ggf. Überschreiben, mit Referenz, ohne Ausgabe,
- 14.9.3.7- Use Case Generieren, ohne Überschreiben, ohne Referenz, mit Ausgabe,
- 14.9.3.8- Use Case Generieren, ohne Überschreiben, mit Referenz, mit Ausgabe,
- 14.9.3.9- Use Case Generieren, ggf. Überschreiben, ohne Referenz, mit Ausgabe und

- 14.9.3.10- Use Case Generieren, ggf. Überschreiben, mit Referenz, mit Ausgabe unterstützen.**[<=]**

A_16818 - (N097.400)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere GENERATE ASYMMETRIC KEY PAIR-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

Das vom Kommando betroffene Schlüsselobjekt wird wie folgt bestimmt:

A_16819 - (N097.500)a K_COS
Wenn der Parameter P2 gleich '00' ist, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

A_16820 - (N097.500)a.1 K_TST
Wenn das Attribut *channelContext.keyReferenceList.signatureCreation* leer ist, dann MUSS es für die funktionale Eignung zulässig sein, wenn das COS dieses Kommandos mit einem beliebigen Trailer beantwortet oder mit einem beliebigen anderen privaten Schlüsselobjekt arbeitet.**[<=]**

G2_N097.500.a.2 - (N097.500)a.2 K_COS
Wenn das Attribute *channelContext.keyReferenceList.signatureCreation* nicht leer ist, gilt: *keyReference* = *keyReferenceList.signatureCreation.keyReference*.**[<=]**

G2_N097.500.b - (N097.500)b K_COS
Falls der Parameter P2 ungleich '00' ist, MUSS*keyReference* = P2 gelten.**[<=]**

G2_N097.500.c - (N097.500)c K_COS
Es wird
affectedObject = SearchSecretKey(
currentFolder,
keyReference,
Wildcard
) gesetzt. Gemäß CosT_a08 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
*Hinweis CosH_1b7: Der Fehler notSupported ist wegen der Wildcard-Suche hier nicht möglich.***[<=]**

G2_N097.600 - (N097.600) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

G2_N097.650 - (N097.650) K_COS
Wenn *operationMode* Element der Menge {'80', '84'} ist und das Attribute *keyAvailable* den Wert True hat, genau dann MUSS das Kommando mit dem Trailer KeyAlreadyPresent terminieren.**[<=]**

G2_N097.700 - (N097.700) K_COS
Wenn *operationMode* Element der Menge {'80', '84', 'C0', 'C4'} ist, dann MÜSSEN folgende Schritte ausgeführt werden:

1. Es MUSS ein Schlüsselpaar (*PrK*, *PuK*) erzeugt werden, dessen Eigenschaften zu den Attributen von *affectedObject* und zu (N002.100) bzw. (N002.500) passen.

2. Anschließend MUSS das Attribute *keyAvailable* mit Transaktionsschutz auf den Wert True geändert werden.

[<=]

G2_N097.800 - (N097.800) K_COS
Wenn *operationMode* Element der Menge {'81'} ist und das Attribute *keyAvailable* den Wert False hat, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.**[<=]**

A_16821 - (N097.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS den Trailer UpdateRetryWarning verwendet, falls ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief.**[<=]**

A_16822 - (N098.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
a. entweder als Trailer MemoryFailure verwendet werden, oder
b. die Kommandobearbeitung gemäß (N031.940) stoppen.**[<=]**

G2_N098.100 - (N098.100) K_COS
Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.**[<=]**

Das DER-TLV codierte Datenobjekt *publicKeyDO* wird wie folgt berechnet:

G2_N098.200.a - (N098.200)a K_COS
Falls *PuK* ein RSA-Schlüssel ist, dann gilt:
1. Setze L_{82} größer gleich OctetLength(*PuK.e*), aber kleiner als 128.
2. Setze *n* = I2OS(*PuK.n*, OctetLength(*PuK.n*)).
3. Setze *e* = I2OS(*PuK.e*, L_{82}).
4. *publicKeyDO* = '7F49 – L_{7F49} – [(81 – L_{81} – *n*) || (82 – L_{82} – *e*)]'.**[<=]**

G2_N098.200.b - (N098.200)b K_COS
Falls *PuK* ein ELC-Schlüssel ist mit dem öffentlichen Punkt *P* und den Domainparametern *dP*, dann gilt *publicKeyDO* = '7F49 – L_{7F49} – (86 – L_{86} – P2OS(*P* , *dP. L*))'.**[<=]**

G2_N098.300.a - (N098.300)a K_COS
Wenn *operationMode* einen Wert aus der Menge {'80', '81', 'C0'} hat, dann MUSS das Datenfeld der Antwortnachricht *publicKeyDO* enthalten. **[<=]**

G2_N098.300.b - (N098.300)b K_COS
Wenn *operationMode* nicht einen Wert aus der Menge {'80', '81', 'C0'} hat, dann MUSS das Datenfeld der Antwortnachricht fehlen.**[<=]**

A_16823 - (N098.400) K_COS
Für die Priorität der Trailer gilt:
a. Die Priorität der Trailer in CosT_6d3 ist herstellerspezifisch.
b. Jeder Trailer in CosT_6d3 MUSS eine höhere Priorität als UpdateRetryWarning haben.
c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.**[<=]**

14.9.4 GET CHALLENGE

Das Kommando GET CHALLENGE erzeugt eine Zufallszahl. Diese steht kartenintern mindestens bei der Ausführung des nächsten Kommandos zur Verfügung. Typischerweise beinhaltet dieses nächste Kommando die Authentisierung einer externen Komponente.

14.9.4.1 Use Case Erzeugen einer Zufallszahl für DES oder RSA Authentisierung

(N098.500) ist absichtlich leer.
(N098.600) ist absichtlich leer.

~~In dieser Variante enthält die APDU des GET CHALLENGE-Kommandos einen Parameter:~~

A_16824 - (N098.500) K_externeWelt {K_Karte}; Option_DES oder Option_RSA_CVC
~~Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich 8 sein.~~**[<=]**

N098.600 - (N098.600) K_externeWelt {K_Karte}, Option_DES oder Option_RSA_CVC
~~Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_31c verwendet werden.~~

Tabelle 310, CosT_31c: GET CHALLENGE für DES oder RSA Authentisierung

	~~Inhalt~~	~~Beschreibung~~
~~CLA~~	~~'00'~~	~~CLA-Byte gemäß [ISO/IEC 7816-4]~~
~~INS~~	~~'84'~~	~~Instruction Byte gemäß [ISO/IEC 7816-4]~~
~~P1~~	~~'00'~~	~~–~~
~~P2~~	~~'00'~~	~~–~~
~~Le~~	~~'08'~~	~~*length*, Anzahl der erwarteten Oktette in den Antwortdaten, hier acht~~

[<=]

14.9.4.2 Use Case Erzeugen einer Zufallszahl für AES oder ELC Authentisierung

In dieser Variante enthält die APDU des GET CHALLENGE-Kommandos einen Parameter:

A_16825 - (N098.620) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich '10' = 16 sein.**[<=]**

A_16826 - (N098.625) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_b6b verwendet werden.

Tabelle 311: CosT_b6b: GET CHALLENGE für AES oder ELC Authentisierung

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'84'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	–
P2	'00'	–
Le	'10'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten, hier 16

[<=]

14.9.4.3 Antwort der Karte auf Erzeugen einer Zufallszahl

Tabelle 312: CosT_652: GET CHALLENGE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*rspData*	Zufallszahl
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	Erfolgreiche Erzeugung einer Zufallszahl

Tabelle 313: CosT_d95: GET CHALLENGE Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
–	–	Derzeit keine Fehlerfälle spezifiziert

Hinweis CosH_144: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16827 - (N098.700) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das GET CHALLENGE-Kommando zusätzliche Trailer verwendet.**[<=]**

14.9.4.4 Kommandoabarbeitung innerhalb der Karte

G2_N098.800.a - (N098.800)a
Das COS MUSS die GET CHALLENGE-Variante aus
~~- 14.9.4.1- Use Case Erzeugen einer Zufallszahl für DES oder RSA Authentisierung und~~
- 14.9.4.2- Use Case Erzeugen einer Zufallszahl für AES oder ELC Authentisierung
unterstützen.**[<=]**

A_16830 - (N098.800)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere GET CHALLENGE-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16831 - (N098.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS im Rahmen der Bearbeitung des GET CHALLENGE-Kommandos

a. Zugriffsregeln auswertet oder
b. als Zugriffsbedingung eines GET CHALLENGE-Kommandos stets ALWAYS verwendet.**[<=]**

G2_N099.000 - (N099.000) K_COS
Es MUSS *rspData* = RAND(Ne) gesetzt werden.**[<=]**

G2_N099.100 - (N099.100) K_COS
Als Trailer MUSS NoError gewählt werden.**[<=]**

G2_N099.200 - (N099.200) K_COS
Das Datenfeld der Antwortnachricht MUSS *rspData* sein.**[<=]**

G2_N099.300 - (N099.300) K_COS
Das Datenfeld der Antwortnachricht *rspData* MUSS zwecks Verwendung in nachfolgenden Kommandos in *RND.ICC* (siehe (N029.900)b) gespeichert werden.**[<=]**

14.9.5 GET RANDOM

Das Kommando GET RANDOM erzeugt eine Zufallszahl. Im Unterschied zu GET CHALLENGE steht diese Zufallszahl nach Abschluss des Kommandos kartenintern nicht für weitere Aktionen zur Verfügung. Dafür erfüllt die mittels GET RANDOM erzeugte Zufallszahl höhere Sicherheitsanforderungen (siehe (N099.356)b). Zudem sind auch andere Längen als 16 Oktette möglich.

14.9.5.1 Use Case Erzeugen kryptographisch sicherer Zufallszahl

In dieser Variante enthält die APDU des GET RANDOM-Kommandos einen Parameter:

A_16832 - (N099.320) K_externeWelt {K_Karte}
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {1, 2, …, 255, WildCardShort} gewählt werden.**[<=]**

A_16833 - (N099.322) K_externeWelt {K_Karte}
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_222 verwendet werden.

Tabelle 314: CosT_222: GET RANDOM

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'84'	Instruction Byte gemäß [ISO/IEC 7816-4] (identisch zu GET CHALLENGE)
P1	'00'	–
P2	'00'	–
Le	*length*	Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.5.2 Antwort der Karte Erzeugen kryptographisch sichere Zufallszahl

Tabelle 315: CosT_086: GET RANDOM Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*random*	Zufallszahl
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	erfolgreiche Erzeugung einer Zufallszahl

Tabelle 316: CosT_211: GET RANDOM Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis CosH_fc0: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16834 - (N099.340) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das GET RANDOM-Kommando zusätzliche Trailer verwendet.**[<=]**

14.9.5.3 Kommandoabarbeitung innerhalb der Karte

G2_N099.344.a - (N099.344)a K_COS
Das COS MUSS die GET RANDOM-Variante aus
- 14.9.5.1- Use Case Erzeugen kryptographisch sicherer Zufallszahl

unterstützen.**[<=]**

A_16835 - (N099.344)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere GET RANDOM-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

G2_N099.348 - (N099.348) K_COS
Als *affectedObject* MUSS *currentFolder* verwendet werden.**[<=]**

G2_N099.352 - (N099.352) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.**[<=]**

A_16836 - (N099.356)a K_COS
Es MUSS *random* = RAND(Ne) gesetzt werden.**[<=]**

A_18226 - (N099.356)b K_PP-COS
Die Güte der Zufallszahl in *random* MUSS im zugehörigen Schutzprofil [PP COS 82] festgelegt werden.~~mindestens [BSI-TR-03116-1#3.5] PTG.3 oder K4-DRNG oder DRG.3 entsprechen.~~**[<=]**

G2_N099.360 - (N099.360) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

G2_N099.364 - (N099.364) K_COS
Das Datenfeld der Antwortnachricht MUSS *random* sein.**[<=]**

14.9.6 GET RESPONSE

A_16837 - (N099.400) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS das GET RESPONSE-Kommando gemäß [ISO/IEC 7816-4]
a. unterstützt oder
b. ablehnt.**[<=]**

14.9.7 LIST PUBLIC KEY

Das Kommando LIST PUBLIC KEY liefert eine Liste der in einer Karte gespeicherten öffentlichen Schlüsselobjekte. Die Ausführung dieses Kommandos ist an keine Vorbedingung geknüpft. Insbesondere ist die Ausführung dieses Kommandos unabhängig vom konkreten Wert von *currentFolder*.

14.9.7.1 Use Case Auslesen der Liste öffentlicher Schlüsselobjekte

In dieser Variante enthält die Liste alle Arten von öffentlichen Schlüsselobjekten.

G2_N099.450 - (N099.450) K_externeWelt {K_Karte}
Die APDU des LIST PUBLIC KEY-Kommandos enthält zwei Parameter.

a. Der Parameter *intendedAction* zeigt an, dass in die Liste alle Arten von öffentlichen Schlüsselobjekten aufzunehmen sind. Der Wert von *intendedAction* MUSS 256 = '0100' sein.
b. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.

[<=]

G2_N099.452 - (N099.452) K_externeWelt {K_Karte}
Es MUSS eine Case 2E-Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2E-Kommando-APDU MÜSSEN die Angaben aus CosT_662 verwendet werden.

Tabelle 317: CosT_662: LIST PUBLIC KEY mit allen Arten öffentlicher Schlüsselobjekte

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier "proprietary" angezeigt
INS	'CA'	Instruction Byte von GET DATA gemäß [ISO/IEC 7816-4]
P1	'0100'	*intendedAction*, gemäß [ISO/IEC 7816-4] proprietärer Wert für GET DATA, hier: Alle Arten von öffentlichen Schlüsselobjekten
P2		
Le	'0000'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.7.2 Antwort der Karte auf Auslesen einer Schlüsselliste

Tabelle 318: CosT_53a: LIST PUBLIC KEY Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx…xx'	*keyReferenceList*	Liste mit Referenzen öffentlicher Schlüsselobjekte
Trailer	**Inhalt**	**Beschreibung**
'62 00'	DataTruncated	Antwortdaten unvollständig
'90 00'	NoError	Erfolgreiche Leseoperation

Tabelle 319: CosT_985: LIST PUBLIC KEY Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
-	-	derzeit ist kein Fehlerfall spezifiziert

Hinweis CosH_04e: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16838 - (N099.458) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das LIST PUBLIC KEY-Kommando zusätzliche Trailer verwendet.**[<=]**

14.9.7.3 Kommandoabarbeitung innerhalb der Karte

G2_N099.460.a - (N099.460)a K_COS
Das COS MUSS die LIST PUBLIC KEY-Variante aus
- 14.9.7.1- Use Case Auslesen der Liste öffentlicher Schlüsselobjekte
unterstützen.**[<=]**

A_16839 - (N099.460)b K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere LIST PUBLIC KEY-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16840 - (N099.461) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS im Rahmen der Bearbeitung des LIST PUBLIC KEY-Kommandos
a. Zugriffsregeln auswertet oder
b. als Zugriffsbedingung eines LIST PUBLIC KEY-Kommandos stets ALWAYS verwendet.**[<=]**

A_16841 - (N099.462) K_COS
Das Datenfeld *keyReferenceList* der Antwortnachricht MUSS durch Ausführen der folgenden Schritte berechnet werden:**[<=]**

A_16842 - (N099.462)a K_COS
Wenn keine Referenz auf ein öffentliches Schlüsselobjekt zurückgemeldet wird, dann MUSS *keyReferenceList* = '' leer sein.**[<=]**

G2_N099.462 - (N099.462)b K_COS
Für jedes öffentliche Schlüsselobjekt in *persistentPublicKeyList* MUSS ein Eintrag in *keyReferenceList* erfolgen.**[<=]**

A_16844 - (N099.462)c K_TST

Für die funktionale Eignung MUSS es zulässig sein, dass für ein öffentliches Schlüsselobjekt in *volatileCache* ein Eintrag in *keyReferenceList*
1. erfolgt oder
2. unterbleibt.**[<=]**

A_16845 - (N099.462)d K_COS
Jeder Eintrag in *keyReferenceList* MUSS ein DER-TLV-Datenobjekt *pukReference* sein.**[<=]**

A_16846 - (N099.462)e K_COS
Alle *pukReference* Datenobjekte MÜSSEN konkateniert werden.**[<=]**

A_16847 - (N099.462)f K_COS
Jedes Datenobjekt *pukReference* MUSS wie folgt aufgebaut sein:

1. Als Tag MUSS 'E0' verwendet werden.
2. Das erste Datenobjekt im DO'E0' MUSS ein Application Identifier DO'4F' sein und die Applikation angeben, in dessen Unterstruktur das öffentliche Schlüsselobjekt gespeichert ist.
3. Wenn das öffentliche Schlüsselobjekt
    i. der im DO'4F' referenzierten Applikation zugeordnet ist, dann DARF DO'E0' KEIN DO'51' enthalten.
    ii. einem Unterordner der im DO'4F'referenzierten Applikation zugeordnet ist, dann MUSS das DO'E0' an zweiter Stelle ein File Reference DO'51' enthalten.
        A. Das Wertfeld des DO'51' MUSS einen Pfad von der Applikation zu dem DF enthalten, welches das öffentliche Schlüsselobjekt enthält.
        B. Die Länge des Wertfeldes von DO'51' MUSS gerade sein.
        C. Die ersten beiden Oktette in DO'51' MÜSSEN den File Identifier eines DF enthalten, welches im Attribut *children* der Applikation enthalten ist.
4. Als nächstes MUSS DO'E0' ein CRT enthalten, dessen Tag gleich
    i. 'A4' ist, falls es sich um ein öffentliches Authentisierungsobjekt handelt.
    ii. 'B6' ist, falls es sich um ein öffentliches Signaturprüfobjekt handelt.
    iii. 'B8' ist, falls es sich um ein öffentliches Verschlüsselungsobjekt handelt.
    iv. Das CRT MUSS ein DO'83' mit dem Attribut *keyIdentifier* des öffentlichen Schlüsselobjektes enthalten.
5. Wenn es für ein öffentliches Schlüsselobjekt mehr als eine Möglichkeit gibt *pukReference* zu codieren, dann MUSS das COS aus der Menge der möglichen Werte genau einen auswählen.

[<=]

G2_N099.463 - (N099.463) K_COS
Das COS MUSS für *keyReferenceList* eine Mindestlänge von *limitRspSecureMessaging* unterstützen. Falls *keyReferenceList* länger ist als vom COS unterstützt, dann

a. MÜSSEN so viele beliebige Einträge wie nötig aus *keyReferenceList* entfernt werden, damit die vom COS unterstützte Länge eingehalten wird.

b. MUSS als Trailer DataTruncated verwendet werden.

[<=]

G2_N099.464.a - (N099.464)a K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

G2_N099.464.b - (N099.464)b K_COS
Für die Priorität der Trailer gilt: DataTruncated MUSS eine höhere Priorität als NoError haben.**[<=]**

A_16848 - (N099.466) K_TST
Es ist unzulässig mehr Einträge aus *keyReferenceList* zu entfernen, als nötig. Im Rahmen funktionaler Zulassungstests MUSS eine Abweichung zur Testerwartungshaltung ausgewiesen werden, falls von allen möglichen Codierungen von *pukReference* aller in *keyReferenceList* fehlender Schlüssel die längste dieser Codierungen zu *keyReferenceList* hinzufügbar ist, ohne dadurch *limitRspSecureMessaging* zu überschreiten.**[<=]**

14.9.8 MANAGE CHANNEL

Das Kommando MANAGE CHANNEL dient dem Öffnen und Schließen von logischen Kanälen mit einer von null verschiedenen Kanalnummer sowie dem Rücksetzen eines beliebigen logischen Kanals. Ob ein Kanal geöffnet, welcher Kanal geschlossen oder zurückgesetzt wird, bestimmen Parameter, die diesem MANAGE CHANNEL-Kommando beigefügt sind.

Es ist möglich, dass ein zusätzlich zum Basiskanal 0 geöffneter logischer Kanal Einfluss auf die Ausführung eines Kommandos hat, wenn dieses Kommando auf Objekte zugreift, welche in anderen logischen Kanälen aktiv sind. Zwecks Vermeidung von Seiteneffekten gilt folgende Anforderung, die von der Kommando-APDU schickenden Einheit einzuhalten ist:

A_16853 - (N099.500) K_externeWelt {K_Karte}
Das Kommando DELETE DARF NICHT an das COS gesendet werden, wenn außer dem Basiskanal noch weitere logische Kanäle geöffnet sind.**[<=]**

14.9.8.1 Use Case Öffnen eines logischen Kanals

In dieser Variante enthält die APDU des MANAGE CHANNEL-Kommandos zwei Parameter:

A_16854 - (N099.504) K_externeWelt {K_Karte}, Option_logische_Kanäle
Der Parameter *intendedAction* zeigt an, dass ein logischer Kanal zu öffnen ist, wobei die Kanalnummer vom COS bestimmt wird. Der Wert von *intendedAction* MUSS 0 = '0000' sein.**[<=]**

A_16855 - (N099.506) K_externeWelt {K_Karte}, Option_logische_Kanäle
Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS 1 = '01' sein.**[<=]**

A_16856 - (N099.508) K_externeWelt {K_Karte}, Option_logische_Kanäle
Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus CosT_42a verwendet werden.

Tabelle 320: CosT_42a: MANAGE CHANNEL zum Öffnen eines logischen Kanals

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'70'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	*intendedAction*, hier Öffnen eines Kanals, Kanalnummer wird vom COS bestimmt.
P2	'00'	
Le	'01'	*length*, Anzahl der erwarteten Oktette in den Antwortdaten

[<=]

14.9.8.2 Use Case Schließen eines logischen Kanals

In dieser Variante enthält die APDU des MANAGE CHANNEL-Kommandos zwei Parameter:

A_16857 - (N099.510) K_externeWelt {K_Karte}, Option_logische_Kanäle
Der Parameter *logicalChannelNumber* MUSS die Nummer eines geöffneten Kanals enthalten, der zu schließen ist. Der Wert von *logicalChannelNumber* MUSS von null verschieden sein und MUSS gemäß [ISO/IEC 7816-4#5.4.1] ins CLA-Byte der APDU eingestellt werden, die an der Schnittstelle "Interface I/O" sichtbar ist.**[<=]**

A_16858 - (N099.512) K_externeWelt {K_Karte}, Option_logische_Kanäle
Der Parameter *intendedAction* zeigt an, dass ein logischer Kanal zu schließen ist, wobei die Kanalnummer im CLA-Byte übertragen wird. Der Wert von *intendedAction* MUSS 32.768 = '8000' sein.**[<=]**

A_16859 - (N099.514) K_externeWelt {K_Karte}, Option_logische_Kanäle
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interface I/O" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_5ca verwendet werden.

Tabelle 321: CosT_5ca: MANAGE CHANNEL zum Schließen eines logischen Kanals

	Inhalt	Beschreibung
CLA	'XX'	CLA-Byte gemäß [ISO/IEC 7816-4] mit einer von 0 verschiedenen Kanalnummer
INS	'70'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'80'	*intendedAction*, hier Schließen eines Kanals, betroffener Kanal wird im CLA-Byte angezeigt.
P2	'00'	

[<=]

14.9.8.3 Use Case Zurücksetzen eines logischen Kanals

In dieser Variante enthält die APDU des MANAGE CHANNEL-Kommandos zwei Parameter:

A_16860 - (N099.520) K_externeWelt {K_Karte}
Der Parameter *logicalChannelNumber* MUSS die Nummer eines geöffneten Kanals enthalten, der zurückzusetzen ist. Der Wert von *logicalChannelNumber* MUSS gemäß

[ISO/IEC 7816-4#5.4.1] ins CLA-Byte der APDU eingestellt werden, die an der Schnittstelle "Interface I/O" sichtbar ist.**[<=]**

A_16861 - (N099.522) K_externeWelt {K_Karte}
Der Parameter *intendedAction* zeigt an, dass ein logischer Kanal zurückzusetzen ist, wobei die Kanalnummer im CLA-Byte übertragen wird. Der Wert von *intendedAction* MUSS 16.384 = '4000' sein.**[<=]**

A_16862 - (N099.524) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interface I/O" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_442 verwendet werden.

Tabelle 322: CosT_442: MANAGE CHANNEL zum Zurücksetzen eines logischen Kanals

	Inhalt	Beschreibung
CLA	'XX'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'70'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'40'	*intendedAction*, hier Zurücksetzen eines Kanals, betroffener Kanal wird im CLA-Byte angezeigt.
P2	'00'	

[<=]

14.9.8.4 Use Case logischer Reset der Applikationsebene

In dieser Variante enthält die APDU des MANAGE CHANNEL-Kommandos einen Parameter:

A_16863 - (N099.530) K_externeWelt {K_Karte}, Option_logische_Kanäle
Der Parameter *intendedAction* zeigt an, dass der Basiskanal zurückzusetzen ist und alle anderen logischen Kanäle zu schließen sind. Der Wert von *intendedAction* MUSS 16.385 = '4001' sein.**[<=]**

A_16864 - (N099.532) K_externeWelt {K_Karte}, Option_logische_Kanäle
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interface I/O" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_901 verwendet werden.

Tabelle 323: CosT_901: MANAGE CHANNEL zum logischen Reset der Applikationsebene

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'70'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'40'	*intendedAction*, hier logischer Reset der Applikationsebene, d.h. Basiskanal zurücksetzen und alle anderen logischen Kanäle schließen.
P2	'01'	

[<=]

14.9.8.5 Antwort der Karte auf Kanalmanagementoperationen

Tabelle 324: CosT_558: MANAGE CHANNEL Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
Kanalnummer	'XX'	Nummer des soeben geöffneten Kanals, oder leer
Trailer	**Inhalt**	**Beschreibung**
'90 00'	NoError	erfolgreiche Operation

Tabelle 325: CosT_ea3: MANAGE CHANNEL Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 81'	NoMoreChannelsAvailable	kein weiterer logischen Kanäle verfügbar

Hinweis CosH_205: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes in CosA_e09 entdeckt wurden.

A_16865 - (N099.540) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das MANAGE CHANNEL-Kommando zusätzliche Trailer verwendet.**[<=]**

14.9.8.6 Kommandoabarbeitung innerhalb der Karte

G2_N099.542.a - (N099.542)a K_COS
Das COS MUSS die MANAGE CHANNEL-Variante aus
- 14.9.8.3- Use Case Zurücksetzen eines logischen Kanals

unterstützen.**[<=]**

G2_N099.542.b - (N099.542)b K_COS, Option_logische_Kanäle
Das COS MUSS die MANAGE CHANNEL-Varianten aus
- 14.9.8.1- Use Case Öffnen eines logischen Kanals,
- 14.9.8.2- Use Case Schließen eines logischen Kanals und
- 14.9.8.4- Use Case logischer Reset der Applikationsebene

unterstützen.**[<=]**

A_16866 - (N099.542)c K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere MANAGE CHANNEL-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16867 - (N099.545) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS im Rahmen der Bearbeitung des MANAGE CHANNEL-Kommandos
a. Zugriffsregeln auswertet oder
b. als Zugriffsbedingung eines MANAGE CHANNEL-Kommandos stets ALWAYS verwendet.**[<=]**

G2_N099.548.a.1 - (N099.548)a.1 K_COS
Wenn *intendedAction* das Öffnen eines logischen Kanals (*indendedAction* = P1P2 = '0000') anzeigt und bereits alle verfügbaren logischen Kanäle geöffnet sind, dann MUSS das Kommando mit dem Trailer NoMoreChannelsAvailable terminieren,**[<=]**

G2_N099.548.a.2 - (N099.548)a.2 K_COS
Wenn *intendedAction* das Öffnen eines logischen Kanals (*indendedAction* = P1P2 = '0000') anzeigt und noch nicht alle verfügbaren logischen Kanäle geöffnet sind, dann MUSS das

COS
i. eine derzeit nicht verwendete Kanalnummer *newChannelNumber* aus dem Intervall [1, 19] allokieren und
ii. einen weiteren logischen Kanal öffnen und
iiii. diesem die allokierte Nummer *newChannelNumber* zuweisen und
iv. dessen Kanalkontext entsprechend (N030.100) initialisieren.**[<=]**

G2_N099.548.a.3 - (N099.548)a.3 K_COS
Wenn *intendedAction* das Öffnen eines logischen Kanals (*indendedAction* = P1P2 = '0000') anzeigt, dann MUSS als Datenfeld der Antwortnachricht I2OS(*newChannelNumber*, 1) verwendet werden.**[<=]**

G2_N099.548.b - (N099.548)b K_COS
Wenn *intendedAction* das Schließen eines logischen Kanals (*indendedAction* = P1P2 = '8000') anzeigt,
1. dann MUSS der entsprechende logische Kanal geschlossen werden und
2. die freiwerdende Kanalnummer MUSS für zukünftige Allokationen verfügbar sein und
3. das Datenfeld der Antwortnachricht MUSS leer sein.**[<=]**

G2_N099.548.c - (N099.548)c K_COS
Wenn *intendedAction* das Zurücksetzen eines logischen Kanals (*indendedAction* = P1P2 = '4000') anzeigt,
1. dann MUSS der *channelContext* des entsprechenden logischen Kanals auf den in (N030.100) definierten Wert gesetzt werden,
2. und das Datenfeld der Antwortnachricht MUSS leer sein.**[<=]**

G2_N099.548.d - (N099.548)d K_COS
Wenn *intendedAction* das logische Resetten (*indendedAction* = P1P2 = '4001') anzeigt
1. dann MÜSSEN bis auf den Basiskanal alle anderen offenen logischen Kanäle geschlossen werden und
2. die freiwerdenden Kanalnummern MÜSSEN für zukünftige Allokationen verfügbar sein, und
3. der *channelContext* des Basiskanals auf den in (N030.100) definierten Wert gesetzt werden, und
4. das Datenfeld der Antwortnachricht MUSS leer sein.**[<=]**

G2_N099.551 - (N099.551) K_COS
Wenn nicht anderweitig spezifiziert, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16868 - (N099.554)a K_TST
Die Priorität der Trailer in CosT_ea3 MUSS herstellerspezifisch sein.**[<=]**

G2_N099.554.b - (N099.554)b K_COS
Jeder Trailer in CosT_ea3 MUSS eine höhere Priorität als NoError haben.**[<=]**

14.9.9 MANAGE SECURITY ENVIRONMENT

Das Kommando MANAGE SECURITY ENVIRONMENT (MSE) verändert im *currentFolder* die Attribute *seIdentifier* und Elemente von *keyReferenceList*. Welche Aktion durchzuführen ist, welches Attribut oder Listenelement betroffen ist und auf welchen Wert sie zu ändern sind, wird durch Parameter bestimmt, die in der Kommandonachricht enthalten sind.

A_16869 - (N099.600) K_externeWelt {K_Karte}
Falls ein symmetrisches Authentisierungsobjekt oder ein symmetrisches Kartenverbindungsobjekt oder ein privates Schlüsselobjekt referenziert wird, dann

besteht der Parameter *keyReference* aus den zwei Teilen *location* und *identifier*. Der Teil *location* zeigt an, ob ein globaler oder DF-spezifischer Schlüssel von der Aktion betroffen ist. Als Wert für *location* MUSS ein Element der Menge {'00', '80'} verwendet werden. Dabei gilt:

c. Der Wert *location* = '00' MUSS verwendet werden, wenn ein globaler Schlüssel betroffen ist.

d. Der Wert *location* = '80' MUSS verwendet werden, wenn ein DF-spezifischer Schlüssel betroffen ist.

e. Der Parameter *identifier* bestimmt das betroffene Schlüsselobjekt. Der Wert von *identifier* MUSS konform zu (N016.400)a, (N017.020)a oder (N017.100)a gewählt werden.

f. Der Parameter *keyReference* MUSS in einem Oktett mit folgendem Wert codiert werden:
keyReference = *location* + *identifier*.

[<=]

14.9.9.1 Use Case Ändern des SE-Identifiers

In dieser Variante enthält die APDU des MSE-Kommandos zwei Parameter:

A_16870 - (N099.700) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'F3' gewählt werden.**[<=]**

A_16871 - (N099.800) K_externeWelt {K_Karte}
Der Parameter *seNo* MUSS gemäß (N007.900)a gewählt werden.**[<=]**

A_16872 - (N099.900) K_externeWelt {K_Karte}
Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus CosT_6fc verwendet werden.

Tabelle 326: CosT_6fc: MSE, Restore Variante

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'F3'	*operationMode* = Auswahl eines SE–Identifiers
P2	'XX'	*seNo* = Wert des auszuwählenden SE–Identifiers

[<=]

14.9.9.2 Use Case Schlüsselauswahl zur internen, symmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16873 - (N100.000) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '41' gewählt werden.**[<=]**

A_16874 - (N100.100) K_externeWelt {K_Karte}

Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'A4' gewählt werden.**[<=]**

A_16875 - (N100.200) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

G2_N100.300.a - (N100.300)a K_externeWelt {K_Karte}
Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß CosT_7a2 gewählt werden, wobei ein Wert aus der Menge {
aesSessionkey4TC,
~~desSessionkey4TC (Option_DES),~~
} verwendet werden MUSS.**[<=]**

A_16876 - (N100.300)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16877 - (N100.400) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_1b4 verwendet werden.

Tabelle 327: CosT_1b4: MSE, Selektion symmetrischer INTERNAL AUTHENTICATE-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'41'	*operationMode* = Setzen eines "internen" Schlüssels *crtTag* = betroffenes Listenelement ist internalAuthenticate
P2	'A4'	
Data	'XX…XX'	'83 – 01 – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.3 Use Case Schlüsselauswahl zur internen, asymmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16878 - (N100.500) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '41' gewählt werden.**[<=]**

A_16879 - (N100.600) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'A4' gewählt werden.**[<=]**

A_16880 - (N100.700) K_externeWelt {K_Karte}

Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

G2_N100.800.a - (N100.800)a K_externeWelt {K_Karte}
Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß CosT_7a2 oder CosT_c40 gewählt werden, wobei ein Wert aus der Menge {
elcAsynchronAdmin,
elcRoleAuthentication,
elcSessionkey4SM,
elcSessionkey4TC,
rsaClientAuthentication,
~~rsaRoleAuthentication (Option_RSA_CVC),~~
~~rsaSessionkey4SM (Option_DES),~~
~~rsaSessionkey4TC (Option_Kryptobox und Option_DES),~~
signPKCS1_V1_5
} verwendet werden MUSS.**[<=]**

A_16881 - (N100.800)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16882 - (N100.900) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_84e verwendet werden.

Tabelle 328: CosT_84e: MSE, Selektion privater INTERNAL AUTHENTICATE-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'41'	*operationMode* = Setzen eines "internen" Schlüssels *crtTag* = betroffenes Listenelement ist internalAuthenticate
P2	'A4'	
Data	'XX…XX'	'84 – 01 – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.4 Use Case Schlüsselauswahl zur externen, symmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16885 - (N101.000) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.**[<=]**

A_16886 - (N101.100) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'A4' gewählt werden.**[<=]**

A_16887 - (N101.200) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

G2_N101.300.a - (N101.300)a K_externeWelt {K_Karte}
Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß CosT_7a2 gewählt werden, wobei ein Wert aus der Menge {
aesSessionkey4TC,
~~desSessionkey4TC (Option_DES),~~
} verwendet werden MUSS.**[<=]**

A_16888 - (N101.300)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16889 - (N101.400) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_a68 verwendet werden.

Tabelle 329: CosT_a68: MSE, Selektion symmetrischer EXTERNAL AUTHENTICATE-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	*operationMode* = Setzen eines "externen" Schlüssels *crtTag* = betroffenes Listenelement ist externalAuthenticate
P2	'A4'	
Data	'XX…XX'	'83 – 01 – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.5 Use Case Schlüsselauswahl zur externen, asymmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16890 - (N101.500) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.**[<=]**

A_16891 - (N101.600) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'A4' gewählt werden.**[<=]**

A_16892 - (N101.700) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N019.500)a gewählt werden.**[<=]**

G2_N101.800.a - (N101.800)a K_externeWelt {K_Karte}
Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß CosT_7a2 gewählt werden, wobei ein Wert aus der Menge {
elcRoleCheck,
~~rsaRoleCheck (Option_RSA_CVC),~~
~~rsaSessionkey4SM (Option_DES),~~
~~rsaSessionkey4TC (OptionKryptobox und Option_DES)~~
} verwendet werden MUSS.
*Hinweis CosH_665: Die Schlüsselselektion für die Algorithmen elcSessionkey4SM und elcSessionkey4TC erfolgt im GENERAL AUTHENTICATE-Kommando, siehe (N085.012).***[<=]**

A_16893 - (N101.800)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16894 - (N101.900) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_b61 verwendet werden.

Tabelle 330: CosT_b61: MSE, Selektion öffentlicher EXTERNAL AUTHENTICATE-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	*operationMode* = Setzen eines öffentlichen Schlüssels *crtTag* = betroffenes Listenelement ist externalAuthenticate
P2	'A4'	
Data	'XX…XX'	'83 – 0C – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.6 Use Case Schlüsselauswahl zur symmetrischen, gegenseitigen Authentisierung

In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16905 - (N102.000) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.**[<=]**

A_16906 - (N102.100) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'A4' gewählt werden.**[<=]**

A_16907 - (N102.200) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

G2_N102.300.a - (N102.300)a K_externeWelt {K_Karte}

Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß CosT_7a2 gewählt werden, wobei ein Wert aus der Menge {
aesSessionkey4SM
~~desSessionkey4SM (Option_DES)~~
} verwendet werden MUSS.**[<=]**

A_16908 - (N102.300)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16909 - (N102.400) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_4f0 verwendet werden.

Tabelle 331: CosT_4f0: MSE, Selektion symmetrischer MUTUAL AUTHENTICATE-Schlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	*operationMode* = Setzen eines symmetrischen Schlüssels *crtTag* = betroffenes Listenelement ist externalAuthenticate
P2	'A4'	
Data	'XX…XX'	'83 – 01 – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.7 Use Case Schlüsselauswahl zur sym. Kartenverbindung ohne Kurvenangabe

In dieser Variante wird ein symmetrisches Kartenverbindungsobjekt gemäß [BSI-TR-03110-3#B.14.1] ausgewählt ohne Referenzierung einer elliptischen Kurve. Anschließend ist es möglich eine Authentisierung mit dem PACE Authentisierungsprotokoll gemäß 15.4.2 durchzuführen. In dieser Variante enthält das MSE-Kommando zwei Parameter:

A_16910 - (N102.440) K_externeWelt {K_Karte}
Der Parameter *OID* bestimmt, welche PACE Variante vom COS verwendet wird und enthält den neuen Wert für das Element *algorithmIdentifier* in den Listenelementen *externalAuthenticate* und *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß CosT_a91 gewählt werden.**[<=]**

A_16911 - (N102.440)a K_externeWelt {K_Karte}
Wenn die Option_kontaktlose_Schnittstelle unterstützt wird, dann MUSS für *OID* ein Wert aus der Menge {
1. id-PACE-ECDH-GM-AES-CBC-CMAC-128,
2. id-PACE-ECDH-GM-AES-CBC-CMAC-192,
3. id-PACE-ECDH-GM-AES-CBC-CMAC-256,
} verwendet werden (siehe 15.4.2).**[<=]**

A_16912 - (N102.440)b K_externeWelt {K_Karte}

Wenn die Option_PACE_PCD unterstützt wird, dann MUSS für *OID* MUSS ein Wert aus der Menge {
1. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128,
2. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192,
3. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256,
} verwendet werden (siehe 15.4.2).**[<=]**

A_16913 - (N102.444) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* in den Listenelementen *externalAuthenticate* und *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

A_16914 - (N102.448) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_a40 verwendet werden.

Tabelle 332: CosT_a40: MSE, Selektion symmetrisches Kartenverbindungsobjekt ohne Kurvenangabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'C1'	*operationMode* = Setzen eines geheimen Schlüsselobjektes *crtTag* = betr. Listenelemente: *externalAuthenticate*, *internalAuthenticate*
P2	'A4'	
Data	'XX…XX'	'80 – I2OS(OctetLength(*OID*), 1) – *OID* || 83 – 01 – *keyRef* '

[<=]

14.9.9.8 Use Case Schlüsselauswahl zur sym. Kartenverbindung mit Kurvenangabe

In dieser Variante wird ein symmetrisches Kartenverbindungsobjekt gemäß [BSI-TR-03110-3#B.14.1] ausgewählt mit Referenzierung einer elliptischen Kurve. Anschließend ist es möglich eine Authentisierung mit dem PACE-Authentisierungsprotokoll gemäß 15.4.2 durchzuführen. In dieser Variante enthält das MSE-Kommando drei Parameter:

A_16915 - (N102.450) K_externeWelt {K_Karte}
Der Parameter *OID* hat dieselbe Bedeutung und Codierung wie in (N102.440).**[<=]**

A_16916 - (N102.452) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* in den Listenelementen *externalAuthenticate* und *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

A_16917 - (N102.454) K_externeWelt {K_Karte}
Der Parameter *idDomainParameter* enthält einen Identifier für Domainparameter der zu verwendenden elliptischen Kurve mit der in CosT_3ce gezeigten Zuordnung gemäß [BSI-TR-03110-3#A.2.1.1], wobei *idDomainparameter* passend zum Attribut *algorithmIdentifier* des referenzierten symmetrischen Kartenverbindungsobjektes gewählt werden MUSS:

Tabelle 333: CosT_3ce: Zuordnung von idDomainParameter zu Domainparameter

ID	Domainparameter	*algorithmIdentifier* im sym. Kartenverbindungsobjekt	
13 = '0D'	brainpoolP256r1	id-PACE-ECDH-GM-AES-CBC-CMAC-128 id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128	oder
16 = '10'	brainpoolP384r1	id-PACE-ECDH-GM-AES-CBC-CMAC-192 id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192	oder
17 = '11'	brainpoolP512r1	id-PACE-ECDH-GM-AES-CBC-CMAC-256 id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256	oder

[<=]

A_16918 - (N102.458) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Es MUSS eine Case 3S-Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3-Kommando-APDU MÜSSEN die Angaben aus CosT_100 verwendet werden.

Tabelle 334: CosT_100: MSE, Selektion symmetrisches Kartenverbindungsobjekt mit Kurvenangabe

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'C1'	*operationMode* = Setzen eines geheimen Schlüsselobjektes *crtTag* = betr. Listenelemente: *externalAuthenticate*, *internalAuthenticate*
P2	'A4'	
Data	'XX…XX'	'80 – I2OS(OctetLength(*OID*), 1) – *OID* ‖ 83 – 01 – *keyRef* ‖ 84 – 01 – *idDomainparameter* '

[<=]

14.9.9.9 Use Case Schlüsselauswahl für Signierschlüssel

Dieser Use Case wird verwendet, um einen Signierschlüssel zu selektieren. Anschließend ist es möglich diesen Schlüssel zu erzeugen oder seinen öffentlichen Teil auszulesen (siehe (N096.600), (N096.652), (N096.900), (N097.200) und (N097.254)) oder mit diesem Schlüssel Signaturen zu erzeugen (siehe (N087.500) und (N088.000)). In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16919 - (N102.500) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '41' gewählt werden.**[<=]**

A_16920 - (N102.600) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'B6' gewählt werden.**[<=]**

A_16921 - (N102.700) K_externeWelt {K_Karte}

Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *signatureCreation*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

G2_N102.800.a - (N102.800)a K_externeWelt {K_Karte}
Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *signatureCreation*. Wert und Codierung MÜSSEN gemäß CosT_7a2 oder CosT_c40 gewählt werden, wobei ein Wert aus der Menge {
rsaClientAuthentication,
sign9796_2_DS2,
signECDSA,
signPKCS1_V1_5,
signPSS
} verwendet werden MUSS.**[<=]**

A_16922 - (N102.800)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16923 - (N102.900) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_d68 verwendet werden.

Tabelle 335: CosT_d68: MSE, Selektion privater Signaturschlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'41'	*operationMode* = Setzen eines privaten Schlüssels *crtTag* = betroffenes Listenelement ist signatureCreation
P2	'B6'	
Data	'XX…XX'	'84 – 01 – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.10 Use Case Schlüsselauswahl zum Prüfen von CV–Zertifikaten

In dieser Variante enthält die APDU des MSE-Kommandos drei Parameter:

A_16924 - (N103.000) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.**[<=]**

A_16925 - (N103.100) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'B6' gewählt werden.**[<=]**

A_16926 - (N103.200) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *verifyCertificate*. Wert und Codierung MÜSSEN gemäß (N019.100)a gewählt werden.**[<=]**

A_16927 - (N103.300) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_25a verwendet werden.

Tabelle 336: CosT_25a: MSE, Selektion öffentlicher Zertifikatsprüfschlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	*operationMode* = Setzen eines öffentlichen Schlüssels *crtTag* = betroffenes Listenelement ist verifyCertificate
P2	'B6'	
Data	'XX…XX'	'83 – 08 – *keyRef* '

[<=]

14.9.9.11 Use Case Schlüsselauswahl zur Datenent- oder Datenumschlüsselung

In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16928 - (N103.400) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '41' gewählt werden.**[<=]**

A_16929 - (N103.500) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'B8' gewählt werden.**[<=]**

A_16930 - (N103.600) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *dataDecipher*. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.**[<=]**

G2_N103.700.a - (N103.700)a K_externeWelt {K_Karte}
Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *dataDecipher*. Wert und Codierung MÜSSEN gemäß CosT_605 gewählt werden, wobei ein Wert aus der Menge {
  elcSharedSecretCalculation,
  rsaDecipherOaep
} verwendet werden MUSS.**[<=]**

A_16931 - (N103.700)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16932 - (N103.800) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_9df verwendet werden.

Tabelle 337: CosT_9df: MSE, Schlüsselselektion zur Ent- und Umschlüsselung

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'41'	*operationMode* = Setzen eines privaten Schlüssels *crtTag* = betroffenes Listenelement ist dataDecipher
P2	'B8'	
Data	'XX…XX'	'84 – 01 – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.12 Use Case Schlüsselauswahl für Verschlüsselung

Dieser Use Case wird verwendet um einen Verschlüsselungsschlüssel zu selektieren. Anschließend ist es möglich diesen Schlüssel zum Verschlüsseln von Daten einzusetzen (siehe Kapitel (N091.424) und (N091.434)). In dieser Variante enthält die APDU des MSE-Kommandos vier Parameter:

A_16933 - (N103.830) K_externeWelt {K_Karte}
Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.**[<=]**

A_16934 - (N103.835) K_externeWelt {K_Karte}
Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'B8' gewählt werden.**[<=]**

A_16935 - (N103.840) K_externeWelt {K_Karte}
Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *dataEncipher*. Wert und Codierung MÜSSEN gemäß (N019.822)a gewählt werden.**[<=]**

G2_N103.845.a - (N103.845)a K_externeWelt {K_Karte}
Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *dataEncipher*. Wert und Codierung MÜSSEN gemäß CosT_605 gewählt werden, wobei ein Wert aus der Menge {
elcSharedSecretCalculation,
rsaEncipherOaep
} verwendet werden MUSS.**[<=]**

A_16936 - (N103.845)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere Werte für *algId*
1. akzeptiert oder
2. ablehnt.**[<=]**

A_16937 - (N103.850) K_externeWelt {K_Karte}
Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle "Interpreter" in CosA_e09 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus CosT_d68 verwendet werden.

Tabelle 338: CosT_d68: MSE, Selektion Verschlüsselungsschlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	*operationMode* = Setzen eines öffentlichen Schlüssels *crtTag* = betroffenes Listenelement ist dataEncipher
P2	'B8'	
Data	'XX…XX'	'83 – 0C – *keyRef* || 80 – 01 – *algId* '

[<=]

14.9.9.13 Antwort der Karte auf Management des Security Environments

Tabelle 339: CosT_33b: MSE Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'90 00'	NoError	Erfolgreiche Übernahme der Kommandodatenparameter

Tabelle 340: CosT_03c: MSE Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegebenen Algorithmus nicht
'6A 88'	KeyNotFound	Zu selektierendes Schlüsselobjekt wurde nicht gefunden

Hinweis CosH_4b8: Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus CosA_e09 entdeckt wurden.

A_16938 - (N103.900) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für das MANAGE SECURITY ENVIRONMENT-Kommando zusätzliche Trailer verwendet.**[<=]**

14.9.9.14 Kommandoabarbeitung innerhalb der Karte

G2_N104.000.a - (N104.000)a K_COS
Das COS MUSS die MANAGE SECURITY ENVIRONMENT-Varianten aus
- 14.9.9.1- Use Case Ändern des SE-Identifiers,
- 14.9.9.3- Use Case Schlüsselauswahl zur internen, asymmetrischen Authentisierung,
- 14.9.9.5- Use Case Schlüsselauswahl zur externen, asymmetrischen Authentisierung,
- 14.9.9.6- Use Case Schlüsselauswahl zur symmetrischen, gegenseitigen Authentisierung,
- 14.9.9.9- Use Case Schlüsselauswahl für Signierschlüssel,
- 14.9.9.10- Use Case Schlüsselauswahl zum Prüfen von CV–Zertifikaten und
- 14.9.9.11- Use Case Schlüsselauswahl zur Datenent- oder Datenumschlüsselung

unterstützen.**[<=]**

G2_N104.000.b - (N104.000)b K_COS, Option_Kryptobox
Das COS MUSS die MANAGE SECURITY ENVIRONMENT-Varianten aus
- 14.9.9.2- Use Case Schlüsselauswahl zur internen, symmetrischen Authentisierung,
- 14.9.9.4- Use Case Schlüsselauswahl zur externen, symmetrischen

Authentisierung und
- 14.9.9.12- Use Case Schlüsselauswahl für Verschlüsselung
unterstützen.**[<=]**

G2_N104.000.c - (N104.000)c K_COS, Option_kontaktlose_Schnittstelle
Das COS MUSS die MANAGE SECURITY ENVIRONMENT-Varianten aus
- 14.9.9.7- Use Case Schlüsselauswahl zur sym. Kartenverbindung ohne Kurvenangabe und
- 14.9.9.8- Use Case Schlüsselauswahl zur sym. Kartenverbindung mit Kurvenangabe
unterstützen.**[<=]**

G2_N104.000.d - (N104.000)d K_COS, Option_PACE_PCD
Das COS MUSS die MANAGE SECURITY ENVIRONMENT-Variante aus
- 14.9.9.7- Use Case Schlüsselauswahl zur sym. Kartenverbindung ohne Kurvenangabe und
- 14.9.9.8- Use Case Schlüsselauswahl zur sym. Kartenverbindung mit Kurvenangabe
unterstützen.**[<=]**

A_16939 - (N104.000)e K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS weitere MANAGE SECURITY ENVIRONMENT-Varianten
1. unterstützt oder
2. ablehnt.**[<=]**

A_16940 - (N104.100) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS im Rahmen der Bearbeitung des MANAGE SECURITY ENVIRONMENT-Kommandos
a. Zugriffsregeln auswertet oder
b. als Zugriffsbedingung eines MANAGE SECURITY ENVIRONMENT-Kommandos stets ALWAYS verwendet.**[<=]**

A_16946 - (N104.200) K_COS
Wenn der Parameter *operationMode* in P1 den Wert 'F3' besitzt, dann MÜSSEN folgende Schritte ausgeführt werden:**[<=]**

G2_N104.200.a - (N104.200)a K_COS
Das COS MUSS in *currentFolder* das Attribut *seIdentifier* (siehe (N030.000)a) auf den Wert des Parameters *seNo* setzen.**[<=]**

G2_N104.200.b - (N104.200)b K_COS
Das COS MUSS jedes Listenelement in *keyReferenceList* (siehe (N029.900)c), welches auf ein Schlüsselobjekt in *currentFolder* zeigt, auf den Wert "leer" setzen.**[<=]**

G2_N104.200.c - (N104.200)c K_COS
Das COS MUSS jedes Element in *globalPasswordList* (siehe (N029.900)i) und in *dfSpecificPasswordList* (siehe (N029.900)j) mittels clearPasswordStatus(…) aus den genannten Listen entfernt werden, wenn es *currentFolder* zugeordnet ist.**[<=]**

G2_N104.200.d - (N104.200)d K_COS
Das COS MUSS den Wert von *currentEF* unverändert lassen.**[<=]**

G2_N104.200.e - (N104.200)e K_COS
Das COS MUSS clearSecurityStatusFolder(*currentFolder*) ausführen.**[<=]**

G2_N104.200.f - (N104.200)f K_COS
Das COS DARF Attribute, die übergeordneten Ordnern zugeordnet sind, NICHT ändern.**[<=]**

A_16947 - (N104.300) K_COS
Wenn der Parameter *operationMode* in P1 einen Wert aus der Menge {'41', '81', 'C1'} besitzt, dann MUSS das COS je nach Länge der Schlüsselreferenz*keyRef* folgende Operationen ausführen:**[<=]**

G2_N104.300.a.1 - (N104.300)a.1 K_COS
Wenn *keyRef* ein Oktett lang ist, dann SOLL mittels SearchKey(*channelContext.currentFolder*, *keyRef*, *algId*) nach dem Schlüssel gesucht werden. Falls diese Funktion
i. den Fehler keyNotFound meldet, dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.
ii. den Fehler notSupported meldet, dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
iii. keinen Fehler meldet, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16948 - (N104.300)a.2 K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass das COS für den Fall einer ein Oktett langen Schlüsselreferenz *keyRef* keine Schlüsselsuche durchführt und dann stets NoError als Trailer verwendet.**[<=]**

A_16949 - (N104.300)b K_COS
Wenn *keyRef* acht Oktett lang ist (öffentliches Signaturprüfobjekt), dann MUSS mittels SearchKey(*channelContext*. *currentFolder*, *keyRef*, verifyCertificate) nach dem Schlüssel gesucht werden.**[<=]**

G2_N104.300.b.1 - (N104.300)b.1 K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N104.300.b.2 - (N104.300)b.2 K_COS
Wenn die Schlüsselsuche den Fehler notSupported meldet, dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N104.300.b.3 - (N104.300)b.3 K_COS
Wenn die Schlüsselsuche keinen Fehler meldet, dann MUSS als Trailer NoError gewählt werden.**[<=]**

A_16950 - (N104.300)c K_COS
Wenn *keyRef* zwölf Oktett lang ist (öffentliches Authentisierungsobjekt, öffentliches Verschlüsselungsobjekt), dann MUSS mittels SearchKey(*channelContext*. *currentFolder*, *keyRef*, *algId*) nach dem Schlüssel gesucht werden.**[<=]**

G2_N104.300.c.1 - (N104.300)c.1 K_COS
Wenn die Schlüsselsuche den Fehler keyNotFound meldet, dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.**[<=]**

G2_N104.300.c.2 - (N104.300)c.2 K_COS
Wenn die Schlüsselsuche den Fehler notSupported meldet, dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.**[<=]**

G2_N104.300.c.3 - (N104.300)c.3 K_COS
Wenn die Schlüsselsuche keinen Fehler meldet, dann MUSS als Trailer NoError gewählt werden.**[<=]**

G2_N104.320.a - (N104.320.)a K_COS
Wenn als Trailer NoError verwendet wird, dann MÜSSEN in *channelContext.keyReferenceList* die durch die Parameter *operationMode* und *crtTag*

gekennzeichneten Listenelemente mit den Parametern aus dem Datenfeld der Kommandonachricht gefüllt werden.**[<=]**

G2_N104.320.b - (N104.320)b K_COS
Wenn als Trailer nicht NoError verwendet wird, dann DARF *channelContext.keyReferenceList* NICHT verändert werden.**[<=]**

(N104.400) Dieser Punkt ist absichtlich leer.

(N104.500) Dieser Punkt ist absichtlich leer.

15 Authentisierungsprotokolle (normativ)

Dieses Kapitel beschreibt, wie eine schlüsselbasierte Authentisierung zwischen einer externen Entität und dem COS abläuft. Die externe Entität wird als Gegenstelle bezeichnet. Aus Symmetriegründen und der leichteren Referenzierbarkeit innerhalb dieses Dokumentes wird eine sprachliche Darstellung gewählt, welche die Gegenstelle als Smartcard mit einem Funktionsumfang beschreibt, der dem Funktionsumfang dieses Dokumentes analog ist. Diese Darstellungsform beschränkt nicht die Allgemeingültigkeit, weil hier lediglich die Schnittstelle zwischen Gegenstelle und COS betrachtet wird, und für die Gegenstelle aus Sicht dieses Dokumentes auch eine beliebige andere Komponente mit entsprechender Funktionalität möglich ist.

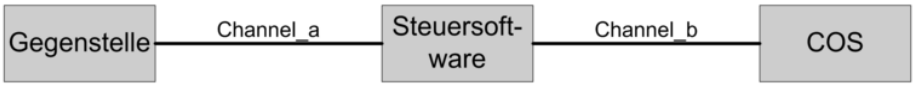


Abbildung 3: CosA_559: Kommunikationsmodell Gegenstelle, Steuersoftware und COS

In diesem Kapitel wird folgendes Kommunikationsmodell verwendet:

- Eine Komponente COS besitze die in diesem Dokument spezifizierten Schnittstelleneigenschaften.
- Eine Komponente Gegenstelle besitze zur Komponente COS analoge Eigenschaften.
- Eine Komponente Steuersoftware besitze eine Ablauflogik. Zudem kommuniziert die Steuersoftware mit der Gegenstelle über den Kommunikationskanal Channel_a und mit dem COS über den Kommunikationskanal Channel_b. Diese beiden Kanäle entsprechen der physikalischen Schnittstelle in CosA_e09.

Jedem der in diesem Kapitel behandelten Authentisierungsverfahren ist ein eigenes Unterkapitel gewidmet. Im Allgemeinen besteht die Authentisierung aus einer Sequenz von einem oder mehreren Kommandos, die über die Kommunikationskanäle Channel_a und Channel_b gesendet werden. Für alle Authentisierungsprotokolle gilt:

A_15905 - (N104.600) K_externeWelt {K_Karte}
Wenn die Sequenz eines Authentisierungsprotokolls aus mehr als einem Kommando-Antwort-Paar besteht (siehe 11.1), so DARF diese Sequenz an der Schnittstelle "Channel_b" (siehe CosA_559) NICHT durch Kommandos unterbrochen werden, welche nicht zu dieser Sequenz gehören.**[<=]**

A_15906 - (N104.700) K_TST
Falls die Anforderung (N104.600) von der externen Entität nicht eingehalten wird, mithin also die Sequenz durch ein unterbrechendes Kommando unterbrochen wird, dann ~~KANN~~ MUSS es für die funktionale Eignung zulässig sein, dass das COS
a. das unterbrechende Kommando akzeptiert, oder
b. das unterbrechende Kommando ablehnt, oder
c. ein Fortsetzen der unterbrochenen Sequenz akzeptiert, oder
d. ein Fortsetzen der unterbrochenen Sequenz ablehnt.**[<=]**

A_15907 - (N104.800) K_COS

Es MUSS möglich sein, dass jede, der hier genannten Sequenzen, ungeschützt (ohne Secure Messaging) übertragen wird. Falls auch die geschützte Übertragung der Sequenz zu unterstützen ist, so ist dies im entsprechenden Unterkapitel vermerkt.**[<=]**

A_15908 - (N104.850) K_externeWelt {K_Karte}
Wenn das erste Kommando dieser Sequenz
a. geschützt übertragen wird, dann MÜSSEN auch alle anderen Kommandos dieser Sequenz geschützt übertragen werden.
b. ungeschützt übertragen wird, dann MÜSSEN auch alle anderen Kommandos dieser Sequenz ungeschützt übertragen werden.**[<=]**

A_15909 - (N104.900) K_TST
Wenn die externe Entität eine der im folgenden genannten Authentisierungssequenzen mit Secure Messaging geschützt überträgt, obwohl die geschützte Übertragung nicht zum normativen Bestandteil gehört, dann MUSS es für die funktionale Eignung zulässig sein, dass das COS dies
a. akzeptiert oder
b. ablehnt.**[<=]**

15.1 Externe Authentisierung

Dieses Unterkapitel behandelt die Authentisierung einer "Gegenstelle" gegenüber dem COS.

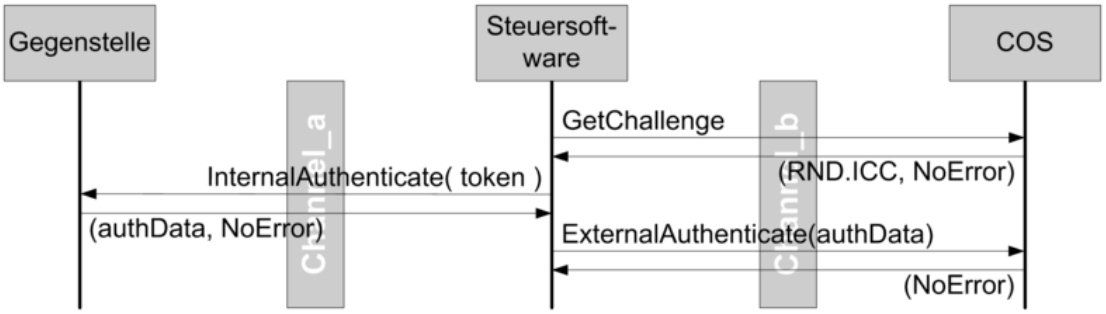


Abbildung 4: CosA_336: Sequenzdiagramm zur externen Authentisierung

Hinweis CosH_fce: Die Bedeutung der Bezeichnungen in CosA_336 ergibt sich aus dem jeweiligen Kontext der Kommandos gemäß Kapitel 14.

15.1.1 Externe Authentisierung mittels symmetrischer Schlüssel

In einer früheren Dokumentenversion wurde in diesem Unterkapitel die einseitige symmetrische Authentisierung ohne Vereinbarung von Sessionkeys behandelt. Weil diese Art der Authentisierung nicht mehr zum normativen Umfang gehört, sind alle Anforderungen dieses Unterkapitels leer.

(N105.000), (N105.100), (N105.200) Diese Punkte sind absichtlich leer.

15.1.2 RSA, asymmetrische Rollenauthentisierung, Option_RSA_CVC

(N105.300), (N105.400), (N105.500) Diese Punkte sind absichtlich leer.

~~Für den Fall, dass eine externe Entität ihre Authentizität mittels eines RSA-Schlüsselpaares nachweisen will und dabei keine Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche CosA_336):~~

G2_N105.300 - (N105.300) K_externeWelt {K_Karte}, Option_RSA_CVC
~~Das erste Kommando der Sequenz MUSS GET CHALLENGE gemäß 14.9.4.1 sein und über Channel_b geschickt werden.~~**[<=]**

G2_N105.400 - (N105.400) K_externeWelt {K_Karte}, Option_RSA_CVC
~~Das zweite Kommando über Channel_b ist das letzte dieser Sequenz und MUSS ein EXTERNAL AUTHENTICATE gemäß 14.7.1.1 mit *algId* gleich rsaRoleCheck sein.~~**[<=]**

G2_N105.500 - (N105.500) K_COS, Option_RSA_CVC
~~Das COS MUSS die (mit Secure Messaging) geschützte Übertragung der Kommandos dieser Sequenz, welche über Channel_b gesendet werden, unterstützen.~~**[<=]**

15.1.3 ELC, asymmetrischer Berechtigungsnachweis

Für den Fall, dass eine externe Entität ihre Authentizität mittels eines ELC-Schlüsselpaares nachweisen will und dabei keine Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche CosA_336):

G2_N105.600 - (N105.600) K_externeWelt {K_Karte}
Das erste Kommando der Sequenz MUSS GET CHALLENGE gemäß (N098.625) sein und über Channel_b geschickt werden.**[<=]**

G2_N105.700 - (N105.700) K_externeWelt {K_Karte}
Das zweite Kommando über Channel_b ist das letzte dieser Sequenz und MUSS ein EXTERNAL AUTHENTICATE gemäß (N083.500) mit *algId* aus der Menge {elcRoleCheck} sein.**[<=]**

G2_N105.800 - (N105.800) K_COS
Das COS MUSS die (mit Secure Messaging) geschützte Übertragung der Kommandos dieser Sequenz, welche über Channel_b gesendet werden, unterstützen.**[<=]**

15.2 Interne Authentisierung

Für den Fall, dass eine externe Entität die Authentizität einer Smartcard mittels eines Schlüssels überprüfen will und dabei keine Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche CosA_e0a):

G2_N105.900 - (N105.900) K_externeWelt {K_Karte}
Das erste Kommando über Channel_b ist das letzte dieser Sequenz für Channel_b und MUSS ein INTERNAL AUTHENTICATE gemäß (N086.400) mit einer *algId* aus folgender Menge sein: {
- elcRoleAuthentication,
- rsaClientAuthentication,
- ~~rsaRoleAuthentication (Option_RSA_CVC),~~

signPKCS1_V1_5
}.**[<=]**

G2_N106.000 - (N106.000) K_COS
Das COS MUSS die (mit Secure Messaging) geschützte Übertragung der Kommandos dieser Sequenz, welche über Channel_b gesendet werden, unterstützen.**[<=]**

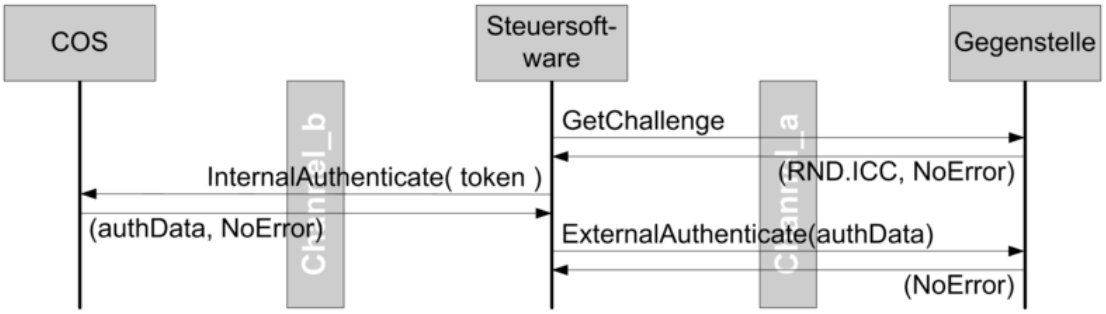


Abbildung 5: CosA_e0a: Sequenzdiagramm zur internen Authentisierung

Hinweis CosH_946: Die Bedeutung der Bezeichnungen in CosA_e0a ergibt sich aus dem jeweiligen Kontext der Kommandos gemäß Kapitel 14.

15.3 Card-2-Card-Authentisierung ohne Sessionkey-Aushandlung

Die Card-2-Card-Authentisierung ohne Sessionkey Aushandlung ist eine spezielle Variante der in 15.1 und 15.2 behandelten Verfahren, wobei die Gegenstelle definitiv ebenfalls eine Smartcard ist.

Falls eine einseitige Authentisierung (nur eine der beiden Komponenten Gegenstelle oder COS authentisiert sich) beabsichtigt ist, wird je nach Richtung der Authentisierung entweder ein Algorithmus aus 15.1 oder aus 15.2 gewählt.

Falls eine gegenseitige Authentisierung (beide Komponenten, also sowohl Gegenstelle als auch COS, authentisieren sich) beabsichtigt ist, wird sowohl ein Algorithmus aus 15.1 als auch ein Algorithmus aus 15.2 gewählt. Typischerweise legen Zugriffsregeln der beteiligten Schlüssel fest, welche Komponente sich zuerst zu authentisieren hat.

15.4 Aushandlung von Sessionkey

Dieses Unterkapitel behandelt die gegenseitige Authentisierung zweier Entitäten, wobei gleichzeitig Sessionkeys ausgehandelt werden.

15.4.1 Sessionkeys mittels symmetrischer Authentisierungsobjekte

Für den Fall, dass eine gegenseitige Authentisierung mittels symmetrischer Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche CosA_a81):

G2_N106.100 - (N106.100) K_externeWelt {K_Karte}

Das erste Kommando der Sequenz MUSS GET CHALLENGE gemäß ~~14.9.4.1~~ (N098.625) sein und über Channel_b zum COSb geschickt werden. Die dabei vom COSb erzeugte Zufallszahl wird mit R1 bezeichnet.**[<=]**

G2_N106.200 - (N106.200) K_externeWelt {K_Karte}
Das zweite Kommando der Sequenz MUSS über Channel_a zum COSa gesendet werden und MUSS ein INTERNAL AUTHENTICATE gemäß (N086.400) mit *algId* gleich symSessionkey4TC sein. Dabei MUSS *cmdData* ~~RND.1 und iccsn8~~ ein *token* (siehe (N086.902)a.4) mit R1 und *iccsn8* des COSb gemäß ~~M2 aus (N084.400)b.1~~ (N084.410)a.6.ii enthalten. Die Antwortdaten des COSa werden mit rspData.2 bezeichnet.**[<=]**

G2_N106.300 - (N106.300) K_externeWelt {K_Karte}
Das dritte Kommando der Sequenz MUSS über Channel_b zum COSb gesendet werden. Es MUSS ein MUTUAL AUTHENTICATE gemäß (N083.800) mit *algId* gleich symSessionkey4SM sein. Als Kommandodaten MUSS rspData.2 verwendet werden. Die Antwortdaten des COSb werden mit rspData.3 bezeichnet.**[<=]**

G2_N106.400 - (N106.400) K_externeWelt {K_Karte}
Das vierte Kommando ist das letzte dieser Sequenz und MUSS über Channel_a zum COSa gesendet werden. Es MUSS ein EXTERNAL AUTHENTICATE gemäß (N083.500) mit *algId* gleich symSessionkey4TC sein. Als Kommandodaten MUSS rspData.3 verwendet werden.**[<=]**

A_15910 - (N106.500) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass sowohl COSa als auch COSb ~~KANN~~ die geschützte Übertragung der Sequenz
a. erlauben oder
b. ablehnen.**[<=]**

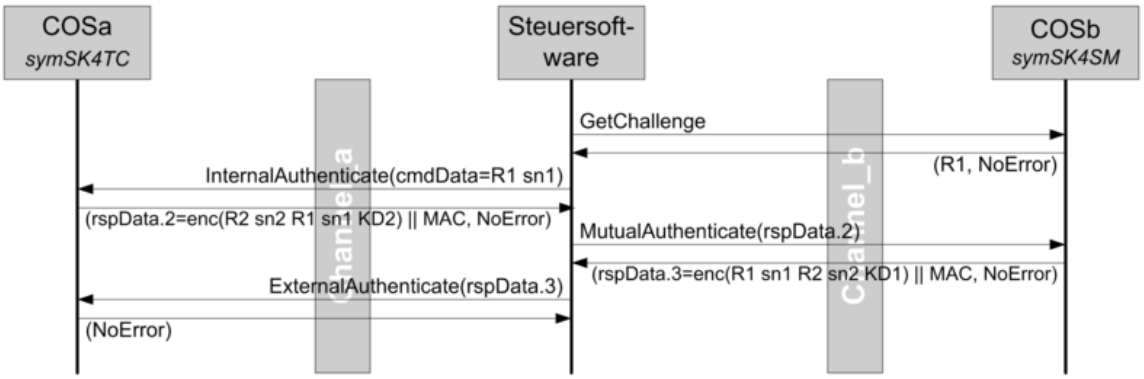


Abbildung 6: CosA_a81: Sequenzdiagramm symmetrische Sessionkey-Aushandlung

Hinweis CosH_399: Aus Gründen der Übersichtlichkeit wurden in Abbildung 6 kürzere Bezeichnungen verwendet, als bei der Beschreibung der Kommandos in Kapitel 14. Die Bedeutung der Bezeichnungen ergibt sich aus dem jeweiligen Kontext.

15.4.2 Sessionkeys mittels symmetrischer Kartenverbindungsobjekte

Für den Fall, dass eine gegenseitige Authentisierung mittels symmetrischer Kartenverbindungsobjekte durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche CosA_8da):

G2_N106.540 - (N106.540) K_COS, Option_kontaktlose_Schnittstelle oder Option_PACE_PCD
Das COS MUSS das PACE-Protokoll in der Version 2 gemäß [BSI-TR-03110-2#3.2] und [BSI-TR-03110-3] für OID aus folgender Menge unterstützen:
a. id-PACE-ECDH-GM-AES-CBC-CMAC-128 mit brainpoolP256r1
b. id-PACE-ECDH-GM-AES-CBC-CMAC-192 mit brainpoolP384r1
c. id-PACE-ECDH-GM-AES-CBC-CMAC-256 mit brainpoolP512r1**[<=]**

G2_N106.545 - (N106.545)a K_COS, Option_kontaktlose_Schnittstelle
Das COS MUSS die PACE-Varianten aus (N106.540) in der Rolle "MRTD Chip (PICC)" unterstützen.**[<=]**

G2_N106.545.b - (N106.545)b K_COS, Option_PACE_PCD
Das COS MUSS die PACE-Varianten aus (N106.540) in der Rolle "Terminal (PCD)" unterstützen.**[<=]**

Hinweis CosH_01a: In (N106.545)b ist absichtlich nur die Option_PACE_PCD enthalten, nicht aber die Option_kontaktlose_Schnittstelle.

A_15911 - (N106.547) K_TST
~~Das COS KANN~~ Für die funktionale Eignung MUSS es zulässig sein, dass das COS die geschützte Übertragung der Sequenz
a. erlaubt oder
b. ablehnt.**[<=]**

G2_N106.550.a - (N106.550)a K_externeWelt {K_Karte}
Im ersten Schritt MUSS zur Komponente COSa ein GENERAL AUTHENTICATE-Kommando gemäß (N085.001) geschickt werden. Die Antwortdaten enthalten das Kryptogramm z, welches im zweiten Schritt auf Seiten der Komponente COSb benötigt wird.**[<=]**

G2_N106.550.b - (N106.550)b K_externeWelt {K_Karte}
Im ersten Schritt MUSS zur Komponente COSb ein GENERAL AUTHENTICATE-Kommando gemäß (N085.031) geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK1_{PCD}$, der im zweiten Schritt auf Seiten der Komponente COSa benötigt wird.**[<=]**

G2_N106.552.a - (N106.552)a K_externeWelt {K_Karte}
Im zweiten Schritt MUSS zur Komponente COSa ein GENERAL AUTHENTICATE-Kommando mit $\sim PK1_{PCD}$ aus (N106.550)b gemäß (N085.003) geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK1_{PICC}$ welcher im dritten Schritt auf Seiten der Komponente COSb benötigt wird.**[<=]**

A_15912 - (N106.552)b K_externeWelt {K_Karte}
Im zweiten Schritt MUSS zur Komponente COSb ein GENERAL AUTHENTICATE-Kommando mit z aus (N106.550)a gemäß (N085.033) geschickt werden. Die Antwortnachricht enthält kein Datenfeld.**[<=]**

G2_N106.554 - (N106.554) K_externeWelt {K_Karte}
Im dritten Schritt MUSS zum COSb ein GENERAL AUTHENTICATE-Kommando mit $\sim PK1_{PICC}$ aus (N106.552)a gemäß (N085.035) geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK2_{PCD}$ welcher im vierten Schritt auf Seiten der Komponente COSa benötigt wird.**[<=]**

G2_N106.556 - (N106.556) K_externeWelt {K_Karte}

Im vierten Schritt MUSS zum COSa ein GENERAL AUTHENTICATE-Kommando mit $\sim PK2_{PCD}$ aus (N106.554) gemäß (N085.005) geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK2_{PICC}$ welcher im fünften Schritt auf Seiten der Komponente COSb benötigt wird.**[<=]**

G2_N106.558 - (N106.558) K_externeWelt {K_Karte}
Im fünften Schritt MUSS zum COSb ein GENERAL AUTHENTICATE-Kommando mit $\sim PK2_{PICC}$ aus (N106.556) gemäß (N085.037) geschickt werden. Die Antwortdaten enthalten einen CMAC T_{PCD} über $\sim PK2_{PICC}$ welcher im sechsten Schritt auf Seiten der Komponente COSa benötigt wird.**[<=]**

G2_N106.560 - (N106.560) K_externeWelt {K_Karte}
Im sechsten Schritt MUSS zum COSa ein GENERAL AUTHENTICATE-Kommando mit T_{PCD} aus (N106.558) gemäß (N085.007) geschickt werden. Die Antwortdaten enthalten einen CMAC T_{PICC} über $\sim PK2_{PCD}$ welcher im siebten Schritt auf Seiten der Komponente COSb benötigt wird.**[<=]**

G2_N106.562 - (N106.562) K_externeWelt {K_Karte}
Im siebten Schritt MUSS zum COSb ein GENERAL AUTHENTICATE-Kommando mit T_{PICC} aus (N106.560) gemäß (N085.039) geschickt werden. Die Antwortnachricht enthält kein Datenfeld.**[<=]**

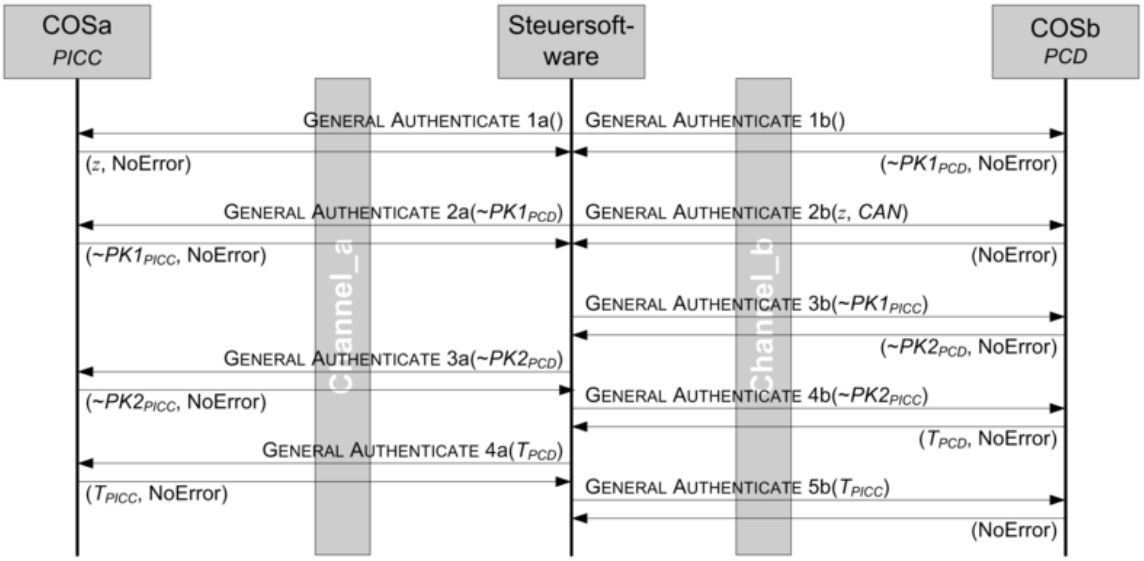


Abbildung 7: CosA_8da: Sequenzdiagramm PACE Authentisierung

Hinweis CosH_c4c: Aus Gründen der Übersichtlichkeit wurden in CosA_8da kürzere Bezeichnungen verwendet, als bei der Beschreibung der Kommandos in 14.7.2. Die Bedeutung der Bezeichnungen ergibt sich aus dem jeweiligen Kontext.

Hinweis CosH_afe: Aus Performanzgründen ist es für die Steuersoftware ratsam die Kommandos des ersten Schrittes zeitlich parallel an die Komponenten schicken. Analog ist es ratsam die Kommandos des zweiten Schrittes zeitlich parallel an die Komponenten zu schicken.

15.4.3 Sessionkeyaushandlung mittels RSA-Schlüssel, Option_DES

Die folgenden Punkte sind absichtlich leer: (N106.600), (N106.700), (N106.800), (N106.900), (N107.000), (N107.100), (N107.200).

Für den Fall, dass eine gegenseitige Authentisierung mittels asymmetrischer RSA-Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche CosA_b19):

G2_N106.600 - (N106.600) K_externeWelt {K_Karte}, Option_DES
Das erste Kommando MUSS zum CosA_ gesendet werden und MUSS GET CHALLENGE gemäß 14.9.4.1 sein. Die dabei vom CosA_ erzeugten Antwortdaten werden mit rspData.1 bezeichnet.**[<=]**

G2_N106.700 - (N106.700) K_externeWelt {K_Karte}, Option_DES
Das zweite Kommando MUSS zum COSb gesendet werden und MUSS ein INTERNAL AUTHENTICATE gemäß 14.7.4.1 mit *algId* gleich rsaSessionkey4SM sein. Dabei MUSS cmdData.2 = rspData.1 || ICCSN8.User gelten mit ICCSN8.User gleich *iccsn8* (siehe (N019.900)c) aus CosA_. Wie die Instanz, welche die Kommando-APDU sendet, Kenntnis von ICCSN8.User erhält, ist nicht Gegenstand dieses Dokumentes. Die korrespondierenden Antwortdaten des COSb werden mit rspData.2 bezeichnet.**[<=]**

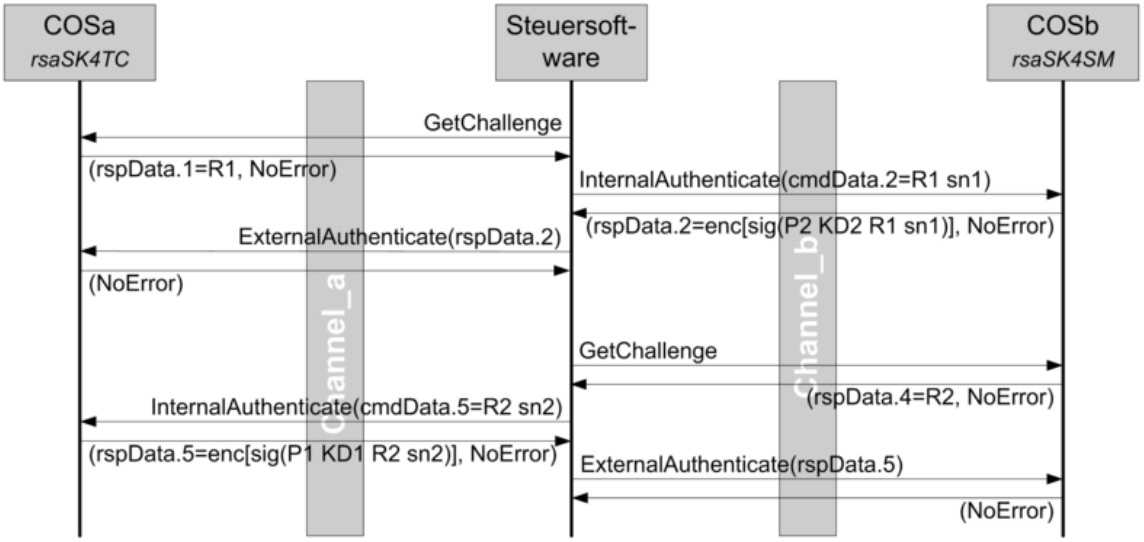
G2_N106.800 - (N106.800) K_externeWelt {K_Karte}, Option_DES
Das dritte Kommando MUSS zum CosA_ gesendet werden. Es MUSS ein EXTERNAL AUTHENTICATE gemäß 14.7.1.1 mit *algId* gleich rsaSessionkey4TC sein. Als Kommandodaten MÜSSEN rspData.2 verwendet werden.**[<=]**

G2_N106.900 - (N106.900) K_externeWelt {K_Karte}, Option_DES
Das vierte Kommando MUSS zum COSb gesendet werden und MUSS GET CHALLENGE gemäß 14.9.4.1 sein. Die dabei vom COSb erzeugten Antwortdaten werden mit rspData.4 bezeichnet.**[<=]**

G2_N107.000 - (N107.000) K_externeWelt {K_Karte}, Option_DES
Das fünfte Kommando MUSS zum CosA_ gesendet werden und MUSS ein INTERNAL AUTHENTICATE gemäß 14.7.4.1 mit *algId* gleich rsaSessionkey4TC sein. Dabei MUSS cmdData.5 = rspData.4 || ICCSN8.COS gelten mit ICCSN8.COS gleich *iccsn8* (siehe (N019.900)c) aus COSb. Wie die Instanz, welche die Kommando-APDU sendet, Kenntnis von ICCSN8.COS erhält, ist nicht Gegenstand dieses Dokumentes. Die korrespondierenden Antwortdaten vom CosA_ werden mit rspData.5 bezeichnet.**[<=]**

G2_N107.100 - (N107.100) K_externeWelt {K_Karte}, Option_DES
Das sechste Kommando ist das letzte der Sequenz und MUSS zum COSb gesendet werden. Es MUSS ein ein EXTERNAL AUTHENTICATE gemäß 14.7.1.1 mit *algId* gleich rsaSessionkey4SM sein. Als Kommandodaten MÜSSEN rspData.5 verwendet werden.**[<=]**

A_15913 - (N107.200) K_TST, Option_DES
Die funktionale Eignung DARF WEDER dem Prüfling CosA_ noch dem Prüfling COSb mit der Begründung verwehrt werden die geschützte Übertragung der Sequenz würde
a. unterstützt
b. nicht unterstützt.**[<=]**



~~**Abbildung 8, CosA_b19: Sequenzdiagramm RSA-Sessionkey-Aushandlung**~~

~~*Hinweis CosH_d37: Aus Gründen der Übersichtlichkeit wurden in CosA_b19 kürzere Bezeichnungen verwendet, als bei der Beschreibung der Kommandos in Kapitel 14. Die Bedeutung der Bezeichnungen ergibt sich aus dem jeweiligen Kontext.*~~

15.4.4 Sessionkeys mittels ELC-Schlüssel

Hinweis CosH_398: Das hier vorgestellte Protokoll geht auf einen Vorschlag zurück, der im DIN NIA 17.4 vorgestellt wurde. Ein Sicherheitsbeweis liegt bislang nicht vor.

Beim hier beschriebenen Authentisierungsprotokoll authentisieren sich zwei Protokollpartner A und B gegenseitig mittels asymmetrischer Kryptographie, die auf elliptischen Kurven basiert und handeln dabei Sessionkeys aus. Die Sessionkeys haben die Verwendungszwecke "Secure Messaging" und "Trusted Channel". Das Protokoll ist so aufgebaut, dass eine erfolgreiche Nutzung der Sessionkeys nur möglich ist, wenn die Verwendungszwecke in den Protokollpartnern unterschiedlich ist. Ohne Beschränkung der Allgemeingültigkeit sei hier angenommen, dass Protokollpartner$A$ die Sessionkeys anschließend für den Zweck "Secure Messaging" nutzt. Dann ist es für eine erfolgreiche Nutzung der Sessionkeys zwingend erforderlich, dass der andere Protokollpartner$B$ die Sessionkeys für den anderen Zweck "Trusted Channel" nutzt. Das Protokoll geht von folgenden Voraussetzungen aus:

1. Beide Protokollpartner A und B verfügen über einen statischen (d. h. persistent gespeicherten) privaten Schlüssel PrK.
2. Die beiden privaten Schlüssel $PrK.a$ und $PrK.b$ verwenden dieselben Domainparameter D.
3. Zu beiden privaten Schlüsseln existiert jeweils ein CV-Zertifikat, welches den korrespondierenden öffentlichen Schlüssel enthält PuK.
4. Die öffentlichen Schlüssel $PuK.a$ und $PuK.b$ werden der jeweiligen Gegenseite bekanntgegeben (d. h. die CV-Zertifikate werden mittels PSO Verify Certificate importiert).
5. In der Komponente A wird der private Schlüssel $PrK.a$ selektiert. Als $algorithmIdentifier$ wird dabei elcSessionkey4SM verwendet.

6. In der Komponente B wird der private Schlüssel $PrK.b$ selektiert. Als $algorithmIdentifier$ wird dabei elcSessionkey4TC verwendet.

Im Rahmen des Authentisierungsprotokolls werden folgende Schritte durchlaufen:

7. Im ersten Schritt wird jede Komponente aufgefordert, den importierten öffentlichen Schlüssel zu selektieren und ein ephemeres Schlüsselpaar zu erzeugen. Die öffentlichen Schlüssel der ephemeren Schlüsselpaare werden aus den Komponenten exportiert. Im Einzelnen:
   a. In Komponente A wird
      i. $PuK.b$ selektiert und
      ii. das ephemere Schlüsselpaar $SK.a$, $PK.a$ generiert und
      iii. $PK.a$ exportiert.
   b. Gleichzeitig wird in der Komponente B
      i. $PuK.a$ selektiert und
      ii. das ephemere Schlüsselpaar $SK.b$, $PK.b$ generiert und
      iii. $PK.b$ exportiert.
8. Im zweiten Schritt werden die öffentlichen Teile der ephemeren Schlüsselpaare in die jeweils andere Komponente importiert und in beiden Komponenten werden je zwei gemeinsame Geheimnisse berechnet und konkateniert. Im Einzelnen:
   a. In Komponente A wird
      i. der ephemere Schlüssel $PK.b$ importiert
      ii. das erste gemeinsame Geheimnis K1 = ECKAvalue($PrK.a$, $PK.b$, D) berechnet.
      iii. das zweite gemeinsame Geheimnis K2 = ECKAvalue($SK.a$, $PuK.b$, D) berechnet.
      iv. Komponente A bildet die Schlüsselableitungsdaten K = K1 || K2.
      v. Der Sicherheitszustand der Komponente A wird entsprechend den Informationen aus dem CV-Zertifikat zu $PuK.b$ gesetzt.
      vi. Mittels der Schlüsselableitungsdaten K werden Sessionkeys berechnet.
   b. Gleichzeitig wird in der Komponente B
      i. der ephemere Schlüssel $PK.a$ importiert
      ii. das erste gemeinsame Geheimnis K1 = ECKAvalue($SK.b$, $PuK.a$, D) berechnet.
      iii. das zweite gemeinsame Geheimnis K2 = ECKAvalue($PrK.b$, $PK.a$, D) berechnet.
      iv. Komponente B bildet die Schlüsselableitungsdaten K = K1 || K2.
      v. Der Sicherheitszustand der Komponente B wird entsprechend den Informationen aus dem CV-Zertifikat zu $PuK.a$ gesetzt.
      vi. Mittels der Schlüsselableitungsdaten K werden Sessionkeys berechnet.

Komponente A bildet K1 mittels des statischen, privaten Schlüssels $PrK.a$ und des ephemeren, öffentlichen Schlüssels $PK.b$. Demgegenüber bildet Komponente B K1 mittels des ephemeren, privaten Schlüssels $SK.b$ und des statischen, öffentlichen Schlüssels $PuK.a$. Diese Asymmetrie ist der Grund, weshalb eine erfolgreiche Etablierung von nutzbaren Sessionkeys nur möglich ist, wenn die $algorithmIdentifier$, welche A und B bei der Selektion der privaten Schlüssel übergeben wurden, anzeigen, dass die Sessionkeys in A und B unterschiedliche Verwendungszwecke haben.

Der Komponente A ist es nur dann mit an Sicherheit grenzender Wahrscheinlichkeit möglich dasselbe K1 wie Komponente B zu berechnen, wenn sie über $PrK.a$ verfügt.

Der Komponente B ist es nur dann mit an Sicherheit grenzender Wahrscheinlichkeit möglich dasselbe K2 wie Komponente A zu berechnen, wenn sie über $PrK.b$ verfügt.

Falls ein Angreifer C entweder Komponente A oder B simuliert, fände trotzdem ein vollständiger Protokolldurchlauf statt. Weder die andere Komponente noch die Steuersoftware haben die Möglichkeit festzustellen, dass anstelle einer authentischen Komponente ein Angreifer am Protokoll teilnimmt. Allerdings wird der Angreifer mit an Sicherheit grenzender Wahrscheinlichkeit andere Sessionkeys berechnen, als der authentische Protokollpartner.

Simuliert der Angreifer die Komponente B, so wird nach dem Protokolldurchlauf in A ein zu B gehörender Sicherheitszustand gesetzt. Da der Angreifer nicht über die korrekten Sessionkeys verfügt, wird dieser Sicherheitszustand in A spätestens mit dem nächsten Kommando entweder wegen (N031.600)a.1 oder wegen (N031.700)a.2.i zurückgesetzt.

Simuliert der Angreifer die Komponente A, so wird nach dem Protokolldurchlauf in B ein zu A gehörender Sicherheitszustand gesetzt. Gemäß der derzeitigen Spezifikationslage wäre es dem Angreifer dann mittels PSO Encipher möglich beliebige Paare von "plaintext" und "ciphertext" erzeugen zu lassen, oder mittels PSO Compute Cryptographic Checksum beliebige Paare von Nachrichten und zugehörigem MAC. Nützlich wären diese Paare nicht. Deshalb wird hier aus Performanzgründen darauf verzichtet, von der Komponente A einen Nachweis über den Besitz der Sessionkeys zu verlangen, bevor in Komponente B der Sicherheitszustand gesetzt wird.

Mit an Sicherheit grenzender Wahrscheinlichkeit sind $PK.a$ und $PK.b$ verschieden, wenn A und B verschieden sind. Falls ein Angreifer C die Kommunikation so manipuliert, dass A identisch zu B ist (technisch wird dabei eine Smartcard auf unterschiedlichen logischen Kanälen angesprochen), dann ist dieser Angriff durch Vergleich von $PK.a$ und $PK.b$ erkennbar.

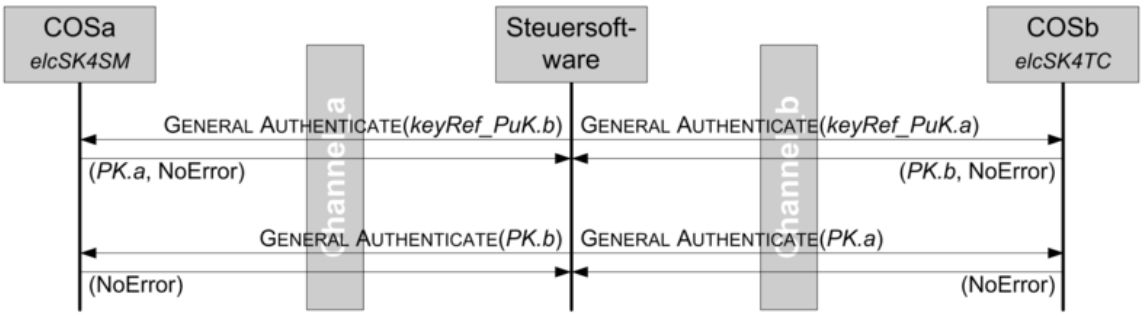


Abbildung 9: CosA_7f8: Sequenzdiagramm ELC-Sessionkey-Aushandlung

Für den Fall, dass eine gegenseitige Authentisierung mittels asymmetrischer ELC-Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche CosA_7f8):

G2_N107.220 - (N107.220) K_externeWelt {K_Karte}
Im ersten Schritt MUSS zu jeder der beiden Komponenten ein GENERAL AUTHENTICATE-Kommando gemäß (N085.012) geschickt werden. Die beiden Antwortdaten enthalten je einen ephemeren, öffentlichen Schlüssel.**[<=]**

G2_N107.230 - (N107.230) K_externeWelt {K_Karte}
Im zweiten und letzten Schritt dieser Sequenz MUSS zu jeder der beiden Komponente ein GENERAL AUTHENTICATE-Kommando gemäß (N085.016) geschickt werden.**[<=]**

A_15914 - (N107.235) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass sowohl COSa als auch COSb ~~KANN~~ die geschützte Übertragung der Sequenz
a. erlauben oder
b. ablehnen.**[<=]**

Hinweis CosH_055: Aus Performanzgründen ist es für die Steuersoftware ratsam die Kommandos des ersten Schrittes zeitlich parallel an die Komponenten schicken. Analog ist es ratsam die Kommandos des zweiten Schrittes zeitlich parallel an die Komponenten zu schicken.

15.5 Statisches Secure Messaging

Dieses Kapitel behandelt die Übertragung von Sessionkeys an ein COS.

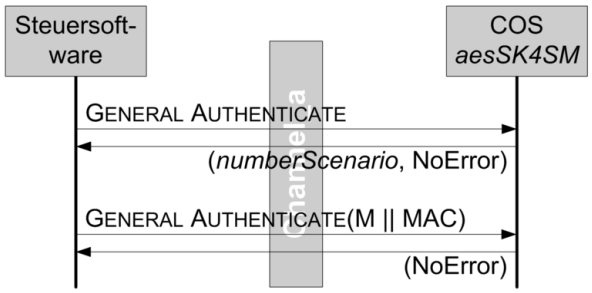


Abbildung 10: CosA_397: Sequenzdiagramm zur Übertragung von Sessionkeys

Hinweis CosH_167: Die Bedeutung der Bezeichnungen in CosA_397 ergibt sich aus dem jeweiligen Kontext der Kommandos gemäß 14.7.2.

Für den Fall, dass eine externe Entität Sessionkeys übertragen will, ist folgende Sequenz zu wählen (vergleiche CosA_397):

G2_N107.250 - (N107.250) K_externeWelt {K_Karte}
Das erste Kommando der Sequenz MUSS über Channel_a geschickt werden und ein GENERAL AUTHENTICATE Kommando
a. gemäß (N085.022) sein, falls der in (N085.051)b beschriebene Fall vorliegt, oder
b. gemäß 14.7.2.5.1 sein, falls der in (N085.051)d beschriebene Fall vorliegt.**[<=]**

G2_N107.252 - (N107.252) K_externeWelt {K_Karte}
Das zweite Kommando über Channel_a ist das letzte dieser Sequenz und MUSS ein GENERAL AUTHENTICATE Kommando
a. gemäß (N085.026) sein, falls der in (N085.051)b beschriebene Fall vorliegt, oder
b. gemäß (N085.041) sein, falls der in (N085.051)d beschriebene Fall vorliegt.**[<=]**

G2_N107.254 - (N107.254) K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass sowohl COSa als auch COSb ~~KANN~~ die geschützte Übertragung der Sequenz

a. erlauben oder
b. ablehnen.**[<=]**

Hinweis CosH_b7c: Hintergründe zu dieser Art der Sessionkeyübertragung finden sich in Kapitel 22.

16 Verschiedenes (normativ)

16.1 Identifier

Tabelle 341: CosT_2a4: Generische AlgorithmIdentifier für Authentisierungszwecke

Name	Oberbegriff für	
asymClientAuthentication		rsaClientAuthentication
asymRoleAuthentication	elcRoleAuthentication	~~rsaRoleAuthentication~~
asymRoleCheck	elcRoleCheck	~~rsaRoleCheck~~
asymSessionkey4SM	elcSessionkey4SM	~~rsaSessionkey4SM~~
asymSessionkey4TC	elcSessionkey4TC	~~rsaSessionkey4TC~~
symSessionkey4SM	aesSessionkey4SM	~~desSessionkey4SM~~
symSessionkey4TC	aesSessionkey4TC	~~desSessionkey4TC~~

Tabelle 342: CosT_7a2: Konkrete AlgorithmIdentifier für Authentisierungszwecke

Name	Codierung	Verwendung
aesSessionkey4SM ~~desSessionkey4SM~~	0101 0100_2 = '54'	Symmetrisch, MUTUAL AUTHENTICATE Sessionkeys für Secure Messaging
aesSessionkey4TC ~~desSessionkey4TC~~	0111 0100_2 = '74'	EXTERNAL AUTHENTICATE, INTERNAL AUTHENTICATE, Sessionkeys für PSO-Kommandos
elcAsynchronAdmin	1111 0100_2 = 'F4'	Asymmetrisch, GENERAL AUTHENTICATE, asynchrone Administration
elcRoleAuthentication	0000 0000_2 = '00'	Asymmetrisch INTERNAL AUTHENTICATE
elcRoleCheck		Asymmetrisch EXTERNAL AUTHENTICATE
elcSessionkey4SM	0101 0100_2 = '54'	Asymmetrisch, GENERAL AUTHENTICATE, Sessionkeys für SecureMessaging
elcSessionkey4TC	1101 0100_2 = 'D4'	Asymmetrisch, GENERAL AUTHENTICATE, Sessionkeys für PSO-Kommandos
rsaClientAuthentication	0000 0101_2 = '05'	INTERNAL AUTHENTICATE
~~rsaRoleAuthentication, Option_RSA_CVC~~	~~0000 0000_2 = '00'~~	~~Asymmetrisch INTERNAL AUTHENTICATE~~
~~rsaRoleCheck, Option_RSA_CVC~~	~~0000 0000_2 = '00'~~	~~Asymmetrisch EXTERNAL AUTHENTICATE~~
~~rsaSessionkey4SM, Option_DES~~	~~0101 0100_2 = '54'~~	~~EXTERNAL AUTHENTICATE, INTERNAL AUTHENTICATE, Sessionkeys für SecureMessaging~~
~~rsaSessionkey4TC, Option_DES~~	~~0111 0100_2 = '74'~~	~~EXTERNAL AUTHENTICATE, INTERNAL AUTHENTICATE, Sessionkeys für PSO-Kommandos~~

Tabelle 343: CosT_605: AlgorithmIdentifier für Ver- und Entschlüsselung

Name	Codierung	Verwendung
aesSessionkey	0000 0000_2 = '00'	PSO Decipher und PSO Encipher
~~desSessionkey, Option_DES~~		
rsaDecipherOaep	1000 0101_2 = '85'	PSO Decipher mit RSA
rsaEncipherOaep	0000 0101_2 = '05'	PSO Encipher und PSO Transcipher mit RSA, rsaDecipherOaep mod 128
elcSharedSecretCalculation	0000 1011_2 = '0B'	PSO Decipher mit ELC

Tabelle 344: CosT_c40: AlgorithmIdentifier für Integrität und Authentizität

Name	Codierung	Verwendung
aesSessionkey	0000 0000_2 = '00'	PSO Compute Cryptographic Checksum und PSO Verify Cryptographic Checksum
~~desSessionkey, Option_DES~~		
sign9796_2_DS2	0000 0111_2 = '07'	PSO Compute Digital Signature
signPKCS1_V1_5	0000 0010_2 = '02'	PSO Compute Digital Signature
signPSS	0000 0101_2 = '05'	PSO Compute Digital Signature
signECDSA	0000 0000_2 = '00'	PSO Compute Digital Signature
verifyCertificate	'XX'	PSO Verify Certificate, da dieser Identifier nie an der physikalischen Schnittstelle verwendet wird, wird er hier nicht festgelegt

Tabelle 345: CosT_a91: Object Identifier, alphabetisch sortiert (informativ)

Name und Codierung	Bemerkung
ansix9p256r1 {1.2.840.10045.3.1.7} '2A8648CE3D030107'	Domainparameter einer elliptischen Kurve gemäß [ANSI X9.62#L.6.4.3]
ansix9p384r1 {1.3.132.0.34} '2B81040022'	Domainparameter einer elliptischen Kurve gemäß [ANSI X9.62#L.6.5.2]
authS_gemSpec-COS-G2_ecc-with-sha256 {1.3.36.3.5.3.1} '2B2403050301'	öffentlicher Schlüssel in einem CV-Zertifikat, Verwendungszweck Authentisierung, SHA_256
authS_gemSpec-COS-G2_ecc-with-sha384 {1.3.36.3.5.3.2} '2B2403050302'	öffentlicher Schlüssel in einem CV-Zertifikat, Verwendungszweck Authentisierung, SHA_384
authS_gemSpec-COS-G2_ecc-with-sha512 {1.3.36.3.5.3.3} '2B2403050303'	öffentlicher Schlüssel in einem CV-Zertifikat, Verwendungszweck Authentisierung, SHA_512
~~authS_ISO9796-2Withrsa_sha256_mutual {1.3.36.3.5.2.4}, Option_RSA_CVC '2B2403050204'~~	~~authentication scheme with RSA signature and DSI according to [ISO/IEC 9796-2] and SHA-256 for mutual authentication with or without establishment of a Trusted Channel~~
brainpoolP256r1 {1.3.36.3.3.2.8.1.1.7} '2B2403030208010107'	Domainparameter einer elliptischen Kurve gemäß [RFC5639#3.4]

brainpoolP384r1 {1.3.36.3.3.2.8.1.1.11} '2B240303020801010b'	Domainparameter einer elliptischen Kurve gemäß [RFC5639#3.6]
brainpoolP512r1 {1.3.36.3.3.2.8.1.1.13} '2B240303020801010d'	Domainparameter einer elliptischen Kurve gemäß [RFC5639#3.7]
ecdsa-with-SHA256 {1.2.840.10045.4.3.2} '2A8648CE3D040302'	öffentlicher Schlüssel in einem CV-Zertifikat, Verwendungszweck Prüfung von CVC, SHA_256
ecdsa-with-SHA384 {1.2.840.10045.4.3.3} '2A8648CE3D040303'	öffentlicher Schlüssel in einem CV-Zertifikat, Verwendungszweck Prüfung von CVC, SHA_384
ecdsa-with-SHA512 {1.2.840.10045.4.3.4} '2A8648CE3D040304'	öffentlicher Schlüssel in einem CV-Zertifikat, Verwendungszweck Prüfung von CVC, SHA_512
id-ELC-shared-secret-calculation	Verschlüsselung gemäß (N004.500)
id-PACE-ECDH-GM-AES-CBC-CMAC-128 {0.4.0.127.0.7.2.2.4.2.2} '04007f00070202040202'	[BSI-TR-03110-3#A.1.1.1] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys für Secure Messaging verwendet werden.
id-PACE-ECDH-GM-AES-CBC-CMAC-192 {0.4.0.127.0.7.2.2.4.2.3} '04007f00070202040203'	[BSI-TR-03110-3#A.1.1.1] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys für Secure Messaging verwendet werden.
id-PACE-ECDH-GM-AES-CBC-CMAC-256 {0.4.0.127.0.7.2.2.4.2.4} '04007f00070202040204'	[BSI-TR-03110-3#A.1.1.1] Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys für Secure Messaging verwendet werden.
id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128 {0.4.0.127.0.7.2.3.4.2.2} '04007f00070203040202'	Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys im Rahmen von PSO-Kommandos verwendet werden.
id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192 {0.4.0.127.0.7.2.3.4.2.3} '04007f00070203040203'	Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys im Rahmen von PSO-Kommandos verwendet werden.
id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256 {0.4.0.127.0.7.2.3.4.2.4} '04007f00070203040204'	Authentisierungsalgorithmus wobei die ausgehandelten Sessionkeys im Rahmen von PSO-Kommandos verwendet werden.
id-RSAES-OAEP {1.2.840.113549.1.1.7} '2A864886F70d010107'	Verschlüsselung gemäß [PKCS#1] Kapitel 7.1.1
oid_cvc_fl_cms {1.2.276.0.76.4.153}	Werte gemäß [gemSpec_OID#Tab_PKI_408], Interpretation

'2A8214004C048119'	der *flagList* gemäß [gemSpec_PKI#Tab_PKI_911]
oid_cvc_fl_ti {1.2.276.0.76.4.152} '2A8214004C048118'	Werte gemäß [gemSpec_OID#Tab_PKI_408], Interpretation der *flagList* gemäß [gemSpec_PKI#Tab_PKI_910]
secp256r1	identisch zu ansix9p256r1, siehe dort
secp384r1	identisch zu ansix9p384r1, siehe dort
~~sigS_ISO9796-2Withrsa_sha256 {1.3.36.3.4.2.2.4}, Option_RSA_CVC '2B240304020204'~~	~~signature scheme with RSA signature and DSI according to [ISO/IEC 9796-2] and SHA-256~~

16.2 Codierungen für Trailer

Tabelle 346: CosT_7aa: Trailer →·Fehlername

Wert	Name	Bedeutung
'62 00'	DataTruncated	Antwortdaten unvollständig
'62 81'	CorruptDataWarning	Die Integrität der Antwortdaten ist nicht gewährleistet
'62 82'	EndOfFileWarning	Es wurden mehr Daten angefordert als die Datei enthält
'62 82'	EndOfRecordWarning	Es wurden mehr Daten angefordert als der Rekord enthält
'62 82'	UnsuccessfulSearch	Pattern wurde in keinem der adressierten Rekords gefunden
'62 83'	FileDeactivated	File, auf welches sich die Operation bezieht, ist deaktiviert
'62 85'	FileTerminated	File, auf welches sich die Operation bezieht, ist terminiert
'62 87'	RecordDeactivated	Rekord, auf welchen sich Operation bezieht, ist deaktiviert
'62 Cx'	TransportStatus	Indikation des Transportschutzverfahrens, siehe Tabelle 144
'62 D0'	PasswordDisabled	Passwortobjekt ausgeschaltet, Verifikation nicht erforderlich
'63 00'	AuthenticationFailure	Authentisierung fehlgeschlagen
'63 CF'	NoAuthentication	Keine Authentisierung mit dem referenzierten Schlüssel
'63 Cx'	RetryCounter	Wert des Fehlbedienungszählers
'63 Cx'	UpdateRetryWarning	Schreibschwierigkeiten
'63 Cx'	WrongSecretWarning	Falsches Passwort in den Kommandodaten
'64 00'	EncipherError	Fehlerhafte Verschlüsselungsoperation
'64 00'	KeyInvalid	Schlüsseldaten fehlen, Generierung erforderlich
'64 00'	ObjectTerminated	Objekt befindet sich im Zustand "Termination state"
'64 00'	ParameterMismatch	Domainparameter passen nicht zusammen
'65 81'	MemoryFailure	Schreibfehler
'67 00'	WrongRecordLength	Falsche Rekordlänge
'68 81'	ChannelClosed	Logischer Kanal nicht geöffnet
'69 81'	NoMoreChannelsAvailable	kein weiterer logischer Kanal verfügbar
'69 81'	VolatileKeyWithoutLCS	volatile Schlüssel werden vom Kommando nicht unterstützt
'69 81'	WrongFileType	Datei unterstützt das aktuelle Kommando nicht
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	CommandBlocked	Rücksetzen des Fehlbedienungszählers nicht mehr möglich
'69 83'	KeyExpired	Der Gültigkeitsbereich des Schlüssels ist abgelaufen
'69 83'	PasswordBlocked	Fehlbedienungszähler abgelaufen

'69 85'	KeyAlreadyPresent	Schlüsseldaten bereits gesetzt, Generierung unmöglich
'69 85'	LongPassword	Neues Passwort zu lang
'69 85'	NoKeyReference	Schlüsselreferenz fehlt, MSE-Set ist notwendig
'69 85'	NoPrkReference	Schlüsselreferenz fehlt, MSE-Set ist notwendig
'69 85'	NoPukReference	Schlüsselreferenz fehlt, MSE-Set ist notwendig
'69 85'	NoRandom	Keine Zufallszahl, GET CHALLENGE ist notwendig
'69 85'	NoRecordLifeCycleStatus	Datei unterstützt das aktuelle Kommando nicht
'69 85'	PasswordNotUsable	Transportschutz aktiv, CHANGE REFERENCE DATA notwendig
'69 85'	WrongRandomLength	Zufallszahl hat falsche Länge, GET CHALLENGE erforderlich
'69 85'	ShortPassword	Neues Passwort zu kurz
'69 86'	NoCurrentEF	Kommandobearbeitung unmöglich, da keine Datei selektiert
'69 88'	IncorrectSmDo	Fehlerhaftes Secure Messaging
'6A 80'	NewFileSizeWrong	*newFileSize* kein Vielfaches der Rekordlänge
'6A 80'	NumberPreconditionWrong	Vorbedingung zum Laden des Scenarios nicht erfüllt
'6A 80'	NumberScenarioWrong	Scenario wurde bereits geladen
'6A 80'	VerificationError	Fehlerhaftes CV-Zertifikat
'6A 80'	WrongCiphertext	Fehlerhaftes Chiffrat
`6A 80'	WrongToken	Token ist fehlerhaft
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegeben Algorithmus nicht
'6A 82'	FileNotFound	Referenzierte Datei nicht gefunden
'6A 83'	RecordNotFound	Referenzierter Rekord nicht verwendbar
'6A 84'	DataTooBig	Zu viele Daten
'6A 84'	FullRecordList	Rekordliste bereits komplett gefüllt
'6A 84'	MessageTooLong	Klartext zu lang für Verschlüsselung
'6A 84'	OutOfMemory	Zu wenig Speicherplatz
'6A 88'	InconsistentKeyReference	Schlüsselreferenz im CV-Zertifikat fehlerhaft
'6A 88'	KeyNotFound	Referenzierten Schlüssel nicht gefunden
'6A 88'	PasswordNotFound	Referenziertes Passwort nicht gefunden
'6A 88'	PrKNotFound	Referenzierten Schlüssel nicht gefunden
'6A 88'	PukNotFound	Referenzierten Schlüssel nicht gefunden

'6A 89'	DuplicatedObject	Neu anzulegendes Objekt existiert bereits
'6A 8A'	DfNameExists	Neu anzulegende Applikation existiert bereits
'6B 00'	OffsetTooBig	Offset zu groß
'6D 00'	InstructionNotSupported	Der im INS-Byte angezeigte Befehl wird nicht unterstützt
'90 00'	NoError	Normale Kommandoausführung, kein Fehler, keine Warnung

17 Hinweise zur Sicherheitsevaluierung (informativ)

Der Hauptteil des Dokumentes enthält im Wesentlichen funktionale Anforderungen. Darüber hinaus sind bei der COS-Entwicklung auch Sicherheitsanforderungen zu beachten. Grundsätzlich sind Sicherheitsanforderungen in einem Protection Profile oder Security Target zu finden. In diesem Anhang werden Aspekte aufgelistet, die im Rahmen einer Evaluierung zu prüfen sind. Die hier genannte Liste erhebt keinen Anspruch auf Vollständigkeit.

1. Gemäß (N034.400) und (N098.800)b ist es funktional zulässig, dass ein COS das Kommando GET CHALLENGE auch für andere, insbesondere sehr kleine Werte für Ne unterstützt. Im Rahmen einer Sicherheitsbetrachtung ist dann nachzuweisen, dass die in *RND.ICC* gespeicherte Zufallszahl nicht weniger zufällig erzeugte Bits enthält, als die in 14.9.4 behandelten Varianten.
2. Dieses Dokument beschreibt in (N001.200) lediglich die funktionalen Anforderungen an einen Zufallszahlengenerator. Die dort erzeugten Zufallszahlen werden für diverse Zwecke eingesetzt. Im Rahmen einer Sicherheitsbetrachtung ist nachzuweisen, dass die Zufallszahlen die in [BSI-TR-03116-1#3.8, 3.9] und im [COS-PP-0082] geforderte Qualität aufweisen.
3. Dieses Dokument beschreibt in 14.9.2 lediglich die funktionalen Anforderungen an die Berechnung eines Fingerprints. Im Rahmen einer Sicherheitsbetrachtung ist nachzuweisen, dass
   a. der Repräsentant M des COS tatsächlich alle Bestandteile des COS umfasst.
   b. bei der Berechnung des Fingerprints gemäß (N096.478) keine Informationen über den Objektcode des COS via Seitenkanäle die Smartcard verlassen.

18 Vorgaben zur Performanz

18.1 Einführung (informativ)

Die Akzeptanz der Smartcards im Gesundheitswesen und der damit verbundenen Abläufe in den medizinischen Einrichtungen (Praxis, Apotheke, Krankenhaus, …) hängt stark von den Zeiten ab, die für einzelne Aktionen benötigt werden. Um im Gesamtsystem zu akzeptablen Werten zu kommen, sind für die einzelnen Komponenten Vorgaben zu treffen, die als zulassungsrelevant in die jeweiligen Spezifikationen aufgenommen werden. In diesem Dokument werden entsprechende Vorgaben für Smartcards im Gesundheitswesen festgelegt.

Es werden sowohl die Randbedingungen als auch die Messparameter festgelegt, um reproduzierbare und aussagekräftige Messungen zu ermöglichen. Es werden Vorgaben für verschiedene Szenarien getroffen. Die ermittelten Messwerte werden über ein Punktesystem bewertet. Die Gesamtpunktzahl setzt sich aus einer gewichteten Addition aller Einzelpunktzahlen zusammen. Das COS wird nur dann von der gematik zugelassen, wenn es eine definierte Mindestpunktzahl (siehe (N109.700)) erreicht.

18.2 Messaufbau (normativ)

Die Bearbeitungszeiten eines Kommandos werden mit dem im Folgenden beschriebenen Messaufbau durchgeführt:



Abbildung 11: CosA_a56: Komponentendiagramm Performanzmessplatz

Die Komponente **Steuersoftware** führt die Performanzmessung durch und protokolliert dabei die Messergebnisse. Diese Komponente besitzt logisch gesehen zwei Kommunikationskanäle zum Interface Device (IFD, Kartenleser). Der eine Kanal transportiert Kommando und Antwort-APDUs, die vom IFD zum Prüfling weitergeleitet werden. Der zweite Kanal dient der Steuerung des IFD sowie dem Auslesen der Laufzeiten, die vom IFD gemessen werden.

Die Komponente **Interface Device (IFD)** transferiert APDUs gemäß der elektrischen Schnittstelle vom und zum Prüfling. Dabei ermittelt das IFD die Bearbeitungszeit eines Kommando-Antwort-APDU-Paares. Wichtig ist dabei, dass das IFD die maximale Übertragungskapazität des Prüflings unterstützt (siehe 18.4).

Die Komponente **"Device under Test" (DuT)** repräsentiert den Prüfling, dessen Performanz zu ermitteln ist.

18.3 Anforderungen an die Steuersoftware (normativ)

A_16953 - (N107.300) K_Performanztest
Die Steuersoftware MUSS in der Lage sein, die einzelnen Schritte der Performanzmessung auszuführen, zu protokollieren und das Gesamtergebnis der Messung aus den einzelnen Resultaten zu ermitteln.**[<=]**

18.4 Anforderungen an das Interface Device (IFD) (normativ)

Bezüglich der Schnittstelle des Interface Devices IFD zum Prüfling DuT gilt:

A_16954 - (N107.350)a K_Performanztest {IFD}
Das IFD MUSS das Übertragungsprotokoll T=1 gemäß 11.2.1 unterstützen.**[<=]**

A_16955 - (N107.350)b K_Performanztest {IFD}
Das IFD SOLL das Übertragungsprotokoll [ISO/IEC 7816-12] gemäß 11.2.2 unterstützen.**[<=]**

A_16956 - (N107.350)c K_Performanztest {IFD}
Das IFD SOLL das kontaktlose Übertragungsprotokoll gemäß 11.2.3 unterstützen.**[<=]**

18.4.1 Anforderungen an das IFD bezüglich T=1

Dieses Unterkapitel enthält die Anforderungen an das IFD bezüglich des Übertragungsprotokolls T=1 gemäß 11.2.1.

A_16959 - (N107.400) K_Performanztest {IFD}
Der Übertragungskanal TPDU_Channel des IFD SOLL die maximale Übertragungskapazität des Prüflings DuT unterstützen. Im Einzelnen bedeutet das:**[<=]**

A_16960 - (N107.500) K_Performanztest {IFD}
Das IFD MUSS im Rahmen einer PPS-Sequenz alle Werte für PPS1 aus der Menge {'18', '95', '96', '97'} unterstützen.

Tabelle 347: CosT_719: Bedeutung PPS1 gemäß [ISO/IEC 7816-3#Table 7 und 8]

PPS1	Teilerfaktor	f_{max} / [MHz]	C / [kBaud]
'18'	372/12 = 31	5	161
'95'	512/16 = 32	5	156
'96'	512/32 = 16	5	313
'97'	512/64 = 8	5	625

[<=]

A_16961 - (N107.600) K_Performanztest {IFD}
Das IFD MUSS eine Versorgungsspannung U_{cc} gemäß [ISO/IEC 7816-3#Table 1] für Class A und Class B liefern.**[<=]**

A_16962 - (N107.700) K_Performanztest {IFD}
Das IFD MUSS einen Versorgungsstrom I_{cc} gemäß [ISO/IEC 7816-3#Table 1] für Class A und Class B liefern.**[<=]**

A_16963 - (N107.800) K_Performanztest {IFD}
Das IFD MUSS während der Performanzmessung ein Clocksignal mit einer Frequenz aus dem Intervall [4,95, 5,05] MHz liefern.**[<=]**

A_16964 - (N107.900) K_Performanztest {IFD}
Das IFD MUSS "direct convention" (siehe [ISO/IEC 7816-3#8.1]) unterstützen.**[<=]**

A_16965 - (N108.000) K_Performanztest {IFD}
Wenn das Byte TC_1 im ATR fehlt, dann MUSS das IFD beim Senden eine "extra guard time" von 12 etu verwenden (siehe [ISO/IEC 7816-3#8.3]).**[<=]**

A_16966 - (N108.100) K_Performanztest {IFD}
Wenn das Byte TC_1 im ATR vorhanden ist, dann MUSS das IFD beim Senden eine "extra guard time" von 11 etu verwenden (siehe [ISO/IEC 7816-3#8.3 und 11.2]).**[<=]**

A_16967 - (N108.200) K_Performanztest {IFD}
Das IFD MUSS beim Empfang eine "character guard time" von 11 etu unterstützen (siehe [ISO/IEC 7816-3#11.2]).**[<=]**

A_16968 - (N108.300) K_Performanztest {IFD}
Das IFD MUSS *BGT* = 22 etu verwenden (siehe [ISO/IEC 7816-3#11.2]).**[<=]**

A_16969 - (N108.400) K_Performanztest {IFD}
Das IFD MUSS für IFSC einen Wert von 254 unterstützen (siehe [ISO/IEC 7816-3#11.4.2]).**[<=]**

A_16970 - (N108.500) K_Performanztest {IFD}
Das IFD MUSS für IFSD einen Wert von 254 unterstützen (siehe [ISO/IEC 7816-3#11.4.2]).**[<=]**

A_16971 - (N108.600) K_Performanztest {IFD}
Das IFD MUSS
a. für eine Kommandonachricht 4096 ~~und SOLL~~ 32.768 als APDU Länge unterstützen.
b. für eine Antwortnachricht 65.638 als APDU Länge unterstützen.**[<=]**

A_18314 - (N108.602) K_Performanztest {IFD}
Das IFD SOLL für eine Kommandonachricht 32.768 als APDU Länge unterstützen.**[<=]**

18.4.2 Anforderungen an das IFD für [ISO/IEC 7816-12] Datenübertragung

Dieses Unterkapitel enthält die Anforderungen an das IFD des Performanztest bezüglich einer Datenübertragung gemäß [ISO/IEC 7816-12].

Hinweis CosH_f60: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

18.4.3 Anforderungen an das IFD bezüglich kontaktloser Datenübertragung

Dieses Unterkapitel enthält die Anforderungen an das IFD des Performanztest bezüglich des kontaktlosen Übertragungsprotokolls gemäß 11.2.3.

Hinweis CosH_31f: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

18.5 Allgemeines (normativ)

18.5.1 Normale Zeitmessung

18.5.1.1 Normale Zeitmessung für das Übertragungsprotokoll T=1

Hier wird die standardmäßig verwendete Zeitmessung für das Übertragungsprotokoll T=1 beschrieben. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

A_16972 - (N108.700) K_Performanztest
Die normale Zeitmessung MUSS die Zeitspanne $t_{Run} = t_{End} – t_{Start} – t_{IO}$ ermitteln, wobei folgende Definitionen gelten:

a. Der Zeitpunkt t_{Start} ist durch den Beginn des ersten Startbits des ersten Characters (siehe [ISO/IEC 7816-3#Figure 7]) der ersten TPDU (siehe [ISO/IEC 7816-3#Figure 23]) gekennzeichnet, welche zur Übertragung einer Kommando-APDU verwendet wird.

b. Der Zeitpunkt t_{End} ist durch das Ende des letzten Paritybits des letzten Characters der letzten TPDU gekennzeichnet, welche zur Übertragung der zugehörigen Antwort-APDU verwendet wird.

c. Die Zeitspanne t_{IO} berücksichtigt die Übertragungszeit von Kommando und Antwort-APDU. Sie MUSS wie folgt berechnet werden:

   i. Sei N_{cmd_APDU} die Anzahl Oktette in der Kommando-APDU, dann gilt für die Anzahl übertragener Oktette N_{cmd_TPDU} auf TPDU-Ebene falls N_{cmd_APDU} größer gleich 1 ist und das Epilogue Field einen "longitudinal redundance code" enthält (siehe [ISO/IEC 7816-3#11.3.4]):
   $$N_{cmd_TPDU} = N_{cmd_APDU} + 8 \text{ ceiling}(N_{cmd_APDU} / 254) – 4.$$

   ii. Für die Anzahl N_{cmd_etu} folgt unter der Voraussetzung, dass pro Oktett 11 etu benötigt werden und eine "block guard time" von 22 etu berücksichtigt wird (siehe [ISO/IEC 7816-3#11.2]):
   $$N_{cmd_etu} = 11\ N_{cmd_TPDU} + 22 \text{ ceiling}(N_{cmd_APDU} / 254).$$

   iii. Analog gilt für die Antwort-APDU:
   $$N_{rsp_TPDU} = N_{rsp_APDU} + 8 \text{ ceiling}(N_{rsp_APDU} / 254) – 4$$
   und
   $$N_{rsp_etu} = 11\ N_{rsp_TPDU} + 22 \text{ floor}(N_{rsp_APDU} / 254).$$

   iv. Daraus folgt: $t_{IO} = (N_{cmd_etu} + N_{rsp_etu})$ Teilerfaktor / f_{max} mit Teilerfaktor und f_{max} in Abhängigkeit von PPS1 aus CosT_719.

[<=]

18.5.1.2 Normale Zeitmessung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12]

Hier wird die standardmäßig verwendete Zeitmessung für Übertragungsprotokoll gemäß [ISO/IEC 7816-12] beschrieben. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

Hinweis CosH_476: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

18.5.1.3 Normale Zeitmessung für die kontaktlose Datenübertragung

Hier wird die standardmäßig verwendete Zeitmessung für die kontaktlose Übertragungsprotokoll beschrieben. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

Hinweis CosH_e03: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

18.5.2 Reguläre Aktivierung der Smartcard

18.5.2.1 Reguläre Aktivierung für das Übertragungsprotokoll T=1

Hier wird die standardmäßige Aktivierung der Karte für das Übertragungsprotokoll T=1 beschrieben, die der Testvorbereitung dient. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

G2_N108.800 - (N108.800) K_Performanztest
Schritt 1: Im Rahmen der regulären Aktivierung der Smartcard MUSS zunächst eine Aktivierung gemäß (N023.920)a und (N023.920)b erfolgen.**[<=]**

G2_N108.900 - (N108.900) K_Performanztest
Schritt 2: Es MUSS eine PPS-Sequenz gemäß [ISO/IEC 7816-3] erfolgen. Die Bits b4 bis b1 in PPS0 MÜSSEN das Übertragungsprotokoll T=1 anzeigen. Als PPS1 MUSS der Wert von TA_1 aus dem ATR verwendet werden.**[<=]**

G2_N109.000 - (N109.000) K_Performanztest
Schritt 3: Das IFD MUSS der Smartcard den Wert von IFSD = 254 präsentieren.**[<=]**

G2_N109.100 - (N109.100) K_Performanztest
Schritt 4: Es MUSS eine beliebige Kommando-APDU bearbeitet werden. Einzige Anforderung an diese Kommando-APDU ist, dass sie so gewählt werden MUSS, dass der Trailer NoError anzeigt und der auszuführende Testfall nicht beeinflusst wird. Hier SOLL eine Selektion von *root* gemäß (N040.800) verwendet werden.**[<=]**

Hinweis CosH_ca9: Die hier beschriebene reguläre Aktivierung hat das Ziel, die Smartcard vollständig zu booten. Typischerweise sind bei aktuellen Smartcard-Betriebssystemen die Initialisierungen so umfangreich, dass sie nicht vollständig innerhalb der Sendezeit des ATR ausführbar sind. Deshalb wird der Smartcard durch das abschließende Kommando genügend Zeit zur Verfügung gestellt, die typischerweise nicht relevant für die Performanzmessung ist.

18.5.2.2 Reguläre Aktivierung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12]

Hier wird die standardmäßige Aktivierung der Karte für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12] beschrieben, die der Testvorbereitung dient. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

Hinweis CosH_2c0: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

18.5.2.3 Reguläre Aktivierung für die kontaktlose Datenübertragung

Hier wird die standardmäßige Aktivierung der Karte für das kontaktlose Übertragungsprotokoll beschrieben, die der Testvorbereitung dient. Es ist möglich, dass hiervon im Einzelfall abgewichen wird. Darauf wird gegebenenfalls explizit hingewiesen.

Hinweis CosH_473: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

18.5.3 Punkteermittlung

Tabelle 348: CosT_508: Definition der Funktion P = points(...)

Input:	$\underline{t}$	n-Tupel mit Zeiten von n Einzelmessungen
	T_R	Referenzzeit
Output:	P	Gewichtete Qualität der Messzeit
Errors:	–	Keine
Notation:		P = points($\underline{t}$, T_R)

Es gelten folgende Definitionen:
(N109.200) Diese Anforderung ist absichtlich leer. Ihr Inhalt wurde nach (N109.450)c verschoben.
(N109.300) Diese Anforderung ist absichtlich leer. Ihr Inhalt wurde nach (N109.450)d verschoben.
(N109.400) Diese Anforderung ist absichtlich leer. Ihr Inhalt wurde nach (N109.450)e verschoben.

A_16978 - (N109.450) K_Performanztest
Der Performanztest MUSS bei der Berechnung der Punktzahl P folgende Formeln verwenden:

$$X = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad = \text{Mittelwert (Erwartungswert) aller Einzelmessungen im } n\text{-Tupel } \underline{t} \quad \text{(a)}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(t_i - X)^2} \qquad = \text{Standardabweichung aller Einzelmessungen im } n\text{-Tupel } \underline{t} \quad \text{(b)}$$

$$f_1 = e^{-\frac{\sigma}{X}} \quad \text{(c)}$$

$$f_2 = 1 - \left(\frac{X}{2 f_1 T_R}\right)^2 \quad \text{(d)}$$

$$P = f_2 \cdot T_R \quad \text{(e)}$$

[<=]

Die Funktion P besteht aus dem Faktor ~~f_1 und~~ f_2 und wird durch T_R gewichtet. Im Faktor f_2 wirkt eine größere Standardabweichung über den Faktor f_1 wie eine verkleinerte Referenzzeit T_R.

Wenn die Standardabweichung null ist, dann ist der Faktor f_1 exakt eins. Größere Standardabweichungen führen zu kleineren Faktoren und damit zu geringeren Punktzahlen P.

Der Faktor f_2 setzt den Mittelwert X in Relation zur Referenzzeit T_R. Deshalb wird f_2 als Funktion von X aufgefasst, die durch T_R parametrisiert wird. Trivialerweise ist es wünschenswert, dass ein kleinerer Wert von X zu einem größeren Wert von f_2 führt. Daraus folgt, dass f_2 streng monoton fallend ist, mithin also die Ableitung von f_2 nach X kleiner null ist. Zudem ist es ratsam, auch die zweite Ableitung von f_2 nach X kleiner gleich null zu wählen, weil es dann eher lohnt, schlechte Mittelwerte zu verbessern, als gute Mittelwerte weiter zu optimieren.

Wenn der Mittelwert X der Messergebnisse gleich dem Referenzwert T_R ist, dann ist der Faktor f_2 gleich 0,75, falls die Standardabweichung vernachlässigbar klein ist.

~~Das Produkt aus f_1 und~~Der Faktor f_2 wird zur Ermittlung des Wertes P mit der Referenzzeit T_R gewichtet. Durch diese Gewichtung korreliert das Gewicht eines Prüfpunktes mit dem Beitrag des Prüfpunkt im Rahmen zusammengesetzter Kommandosequenzen.

18.5.4 Gesamtbewertung

Die Gesamtbewertung wird durch gewichtetes Addieren der Einzelpunktzahlen ermittelt. Die Gewichte korrelieren mit der Häufigkeit des Auftretens der Einzelpositionen im Feld.

A_16979 - (N109.460) K_Performanztest
Der Performanztest MUSS bei der Bewertung der einzelnen Prüfpunkte für das Basisbetriebssystem die in CosT_1b9 dargestellten Werte für die Referenzzeit T_{Ri} und die Gewichte g_i zugrundelegen.

Tabelle 349: CosT_1b9: Gesamtbewertung für das Basisbetriebssystem

Prüfpunkt	Use Case	Kapitel	Bezeichnung	T_{Ri} / [ms]	g_i	g_i T_{Ri} / [ms]
Kanalkapazität	(N024.100)	18.6.1	P_{IO}	17	50.000	850.000
Karte starten	(N023.920)	18.7	$P_{KarteStarten}$	50	2.000	100.000
ACTIVATE Datei	(N034.800)	18.8.1	$P_{activate_EF}$	30	100	3.000
ACTIVATE Ordner			$P_{activate_DF}$	30	100	3.000
ACTIVATE private Key	(N034.814)		$P_{activate_PrK}$	30	100	3.000
ACTIVATE sym. Key			$P_{activate_SK}$	30	100	3.000
ACTIVATE PuK	(N034.824)		$P_{activate_PuK}$	30	100	3.000
ACTIVATE Pwd	(N034.834)		$P_{activate_Pwd}$	30	100	3.000
DEACTIVATE Datei	(N036.000)		$P_{deactivate_EF}$	30	100	3.000
DEACTIVATE Ordner			$P_{deactivate_DF}$	30	100	3.000
DEACTIVATE private Key	(N036.014)		$P_{deactivate_PrK}$	30	100	3.000
DEACTIVATE sym. Key			$P_{deactivate_SK}$	30	100	3.000
DEACTIVATE PuK	(N036.024)		$P_{deactivate_PuK}$	30	100	3.000
DEACTIVATE Pwd	(N036.034)		$P_{deactivate_Pwd}$	30	100	3.000
DELETE Datei	(N037.100)		P_{delete_EF}	100	30	3.000
DELETE Ordner			P_{delete_DF}	500	10	5.000

DELETE private Key	(N037.114)		P_{delete_PrK}	60	50	3.000
DELETE sym. Key			P_{delete_SK}	60	50	3.000
DELETE PuK	(N037.124)		P_{delete_PuK}	60	50	3.000
DELETE Pwd	(N037.134)		P_{delete_Pwd}	60	50	3.000
LOAD APPLICATION	14.2.5		$P_{LoadApp}$	1.000	10	10.000
SELECT Datei	(N046.700)	18.8.2	P_{select_EF}	10	100.000	1.000.000
SELECT Ordner	(N044.900)	18.8.17	P_{select_DF}	10	25.000	250.000
TERMINATE CARD USAGE	(N048.700)	18.8.4	$P_{terminateCard}$	30	100	3.000
TERMINATE DF	(N048.800)	18.8.1	$P_{terminate_DF}$	30	100	3.000
TERMINATE EF	(N048.903)		$P_{terminate_EF}$	30	100	3.000
TERMINATE privateKey	(N048.914)		$P_{terminate_PrK}$	30	100	3.000
TERMINATE sym. Key			$P_{terminate_SK}$	30	100	3.000
TERMINATE PuK	(N048.924)		$P_{terminate_PuK}$	30	100	3.000
TERMINATE Pwd	(N048.934)		$P_{terminate_Pwd}$	30	100	3.000
SET LOGICAL EOF	(N052.932)	18.8.5	P_{SetEOF}	600	100	60.000
ERASE BINARY wipe	(N049.100)	18.8.6	$P_{WipeBin}$	600	100	60.000
READ BINARY *b*	(N051.100)	18.8.7	$P_{ReadBinary,b}$	11	100.000	1.100.000
READ BINARY *m*			$P_{ReadBinary,m}$	5	200.000	1.000.000
UPDATE BINARY *b*	(N053.200)	18.8.6	$P_{UpdateBin,b}$	60	5.000	300.000
UPDATE BINARY *m*			$P_{UpdateBin,m}$	60	5.000	300.000
WRITE BINARY *b*	(N055.205)	18.8.5	$P_{wirte,b}$	60	100	6.000
WRITE BINARY *m*			$P_{wirte,m}$	60	100	6.000
ACTIVATE RECORD	(N055.500)	18.8.8	$P_{ActivateRec}$	30	100	3.000

APPEND RECORD *b*	(N058.400)		$P_{AppendRecord,b}$	60	8.000	480.000
APPEND RECORD *m*			$P_{AppendRecord,m}$	40	10.000	400.000
DEACTIVATE RECORD	(N060.700)		$P_{DeactivateRec}$	30	100	3.000
DELETE RECORD	(N063.422)		$P_{DeleteRec}$	50	100	5.000
ERASE RECORD wipe *b*	(N063.600)		$P_{WipeRecord,b}$	30	100	3.000
ERASE RECORD wipe *m*			$P_{WipeRecord,m}$	30	100	3.000
READ RECORD *b*	(N065.700)		$P_{ReadRecord,b}$	8	150.000	1.200.000
READ RECORD *m*			$P_{ReadRecord,m}$	4	300.000	1.200.000
SEARCH RECORD	(N067.900)	18.8.9	$P_{SearchRec}$	120	500	60.000
UPDATE RECORD *b*	(N070.300)	18.8.8	$P_{UpdateRecord,b}$	30	5.000	150.000
UPDATE RECORD *m*			$P_{UpdateRecord,m}$	40	5.000	200.000
CHANGE REFERENCE DATA change	(N073.300)	18.8.1	$P_{ChRefData}$	90	250	22.500
CHANGE REFERENCE DATA set	(N073.700)		P_{SetPIN}	50	50	2.500
DISABLE VERIFICATION REQUIREMENT	(N075.386)		$P_{DisablePIN}$	50	50	2.500
ENABLE VERIFICATION REQUIREMENT	(N078.586)		$P_{EnablePIN}$	50	50	2.500
GET PIN STATUS	(N077.900)		$P_{GetPinStatus}$	10	50	500
RESET RETRY COUNTER	(N079.300)		$P_{ResetRC}$	70	200	14.000
VERIFY	(N082.200)		P_{VERIFY}	60	1.000	60.000
EXTERNAL AUTHENTICATE	(N083.500)	18.8.11.1	$P_{RoleCheck,ELC256}$	150	1.400	210.000
		18.8.11.2	$P_{RoleCheck,ELC384}$	200	100	20.000
		18.8.11.3	$P_{RoleCheck,ELC512}$	300	50	15.000
GET SECURITY STATUS KEY	(N085.444)	18.8.11.1	$P_{SesKey,ELC256}$	10	100	1.000
INTERNAL AUTHENTICATE	(N086.400)	18.8.12	$P_{RoleAuth,ELC256}$	120	1.400	168.000

			$P_{RoleAuth,ELC384}$	190	100	19.000
			$P_{RoleAuth,ELC512}$	280	50	14.000
Sessionkeyaushandlung für Secure Messaging	15.4.1	18.8.10	$P_{SK4SM,AES128}$	140	800	112.000
			$P_{SK4SM,AES192}$	160	500	80.000
			$P_{SK4SM,AES25}$	180	100	18.000
	15.4.4	18.8.11.1	$P_{SesKey,ELC256}$	400	1.000	400.000
		18.8.11.2	$P_{SesKey,ELC384}$	600	500	300.000
		18.8.11.3	$P_{SesKey,ELC512}$	900	100	90.000
PSO Compute Digital Signature	(N085.500)	18.8.13	$P_{signPSS,2048}$	200	1.000	200.000
			$P_{signPSS,3072}$	750	10	7.500
		18.8.14	$P_{signECDSA,256}$	100	2.000	200.000
			$P_{signECDSA,384}$	150	600	90.000
			$P_{signECDSA,512}$	250	100	25.000
PSO Decipher	(N089.200)	18.8.15	$P_{dec,2048}$	200	2.000	400.000
			$P_{dec,3072}$	900	100	90.000
	(N089.800)	18.8.16	$P_{dec,256}$	150	2.000	300.000
			$P_{dec,384}$	180	600	108.000
			$P_{dec,512}$	270	100	27.000
PSO Encipher	(N090.790)	18.8.15	$P_{enc,2048}$	40	200	8.000
	(N091.400)	18.8.16	$P_{enc,256}$	200	1.000	200.000
			$P_{enc,384}$	300	500	150.000
			$P_{enc,512}$	500	100	50.000
PSO Verify Certificate	(N095.410)	18.8.11.1	$P_{Import,ELC256}$	300	500	150.000
		18.8.11.2	$P_{Import,ELC384}$	630	100	63.000

		18.8.11.3	$P_{Import,ELC512}$	900	20	18.000
PSO Verify Digital Signature	(N096.388)	18.8.14	$P_{verifyECDSA,256}$	80	12	960
			$P_{verifyECDSA,384}$	140	6	840
			$P_{verifyECDSA,512}$	220	1	220
FINGERPRINT	(N096.454)	18.8.3	$P_{fingerprint}$	8.000	1	8.000
GENERATE ASYMMETRIC KEY PAIR	(N097.266)	18.8.16	$P_{GAKP,256}$	140	500	70.000
			$P_{GAKP,384}$	200	100	20.000
			$P_{GAKP,512}$	280	30	8.400
GET CHALLENGE	(N098.625)	18.8.17	$P_{challenge}$	10	1.000	10.000
MANAGE CHANNEL reset Channel	(N099.524)		P_{reset_Ch}	5	500	2.500
MANAGE SECURITY ENVIRONMENT Restore	(N099.900)	18.8.18	$P_{MSE_Restore}$	5	500	2.500
MANAGE SECURITY ENVIRONMENT Set	14.9.9		P_{MSE_Set}	10	10.000	100.000
GET RANDOM	(N099.322	18.8.22	$P_{Random,b}$	4	18	72
			$P_{Random,m}$	40	2	80
Spaltensummen				23.589	1.000.000	12.687.572

[<=]

A_16980 - (N109.465) K_Performanztest

Der Performanztest MUSS bei der Bewertung der einzelnen Prüfpunkte für die Option_kontaktlose_Schnittstelle die in CosT_930 dargestellten Werte für die Referenzzeit T_{Ri} und die Gewichte g_i zugrundelegen.

Tabelle 350: CosT_930: Gesamtbewertung für Option_kontaktlose_Schnittstelle

Prüfpunkt	Use Case	Kapitel	Bezeichnung	T_{Ri} / [ms]	g_i	g_i T_{Ri} / [ms]
GENERAL AUTHENTICATE	14.7.2.1	18.8.19	P_{PACE}	750	500	375.000
Spaltensummen				750	500	375.000

[<=]

A_16981 - (N109.470) K_Performanztest

Der Performanztest MUSS bei der Bewertung der einzelnen Prüfpunkte für die Option_Kryptobox die in CosT_005 dargestellten Werte für die Referenzzeit T_{Ri} und die Gewichte g_i zugrundelegen.

Tabelle 351: CosT_005: Gesamtbewertung für Option_Kryptobox

Prüfpunkt	Use Case	Kapitel	Bezeichnung	T_{Ri} / [ms]	g_i	$g_i\ T_{Ri}$ / [ms]
Sessionkeyaushandlung für Trusted Channel	15.4.1	18.8.20	$P_{SK4TC,AES128}$	70	1.400	98.000
			$P_{SK4TC,AES192}$	90	500	450.000
			$P_{SK4TC,AES256}$	100	100	10.000
PSO Compute Cryptographic Checksum	(N087.228)	18.8.21	$P_{compute128,b}$	10	10.000	100.000
			$P_{compute128,m}$	5	22.000	110.000
			$P_{compute192,b}$	10	6.000	60.000
			$P_{compute192,m}$	6	10.000	60.000
			$P_{compute256,b}$	10	500	5.000
			$P_{compute256,m}$	7	1.000	7.000
PSO Decipher	(N089.845)		$P_{dec128,b}$	10	10.000	100.000
			$P_{dec128,m}$	10	22.000	220.000
			$P_{dec192,b}$	10	6.000	60.000
			$P_{dec192,m}$	12	10.000	120.000
			$P_{dec256,b}$	10	500	5.000
			$P_{dec256,m}$	14	1.000	14.000
PSO Encipher	(N091.446)		$P_{enc128,b}$	10	10.000	100.000
			$P_{enc128,m}$	10	22.000	220.000
			$P_{enc192,b}$	10	6.000	60.000
			$P_{enc192,m}$	12	10.000	120.000
			$P_{enc256,b}$	10	500	5.000

			$P_{enc256,m}$	14	1.000	14.000
PSO Verify Cryptographic Checksum	(N096.346)		$P_{verify128,b}$	10	10.000	100.000
			$P_{verify128,m}$	5	22.000	110.000
			$P_{verify192,b}$	10	6.000	60.000
			$P_{verify192,m}$	6	10.000	60.000
			$P_{verify256,b}$	10	500	5.000
			$P_{verify256,m}$	7	1.000	7.000
Spaltensummen				488	200.000	1.875.000

[<=]

A_16982 - (N109.475) K_Performanztest

Der Performanztest MUSS bei der Bewertung der einzelnen Prüfpunkte für die Option_logische_Kanäle die in CosT_51e dargestellten Werte für die Referenzzeit T_{Ri} und die Gewichte g_i zugrundelegen.

Tabelle 352: CosT_51e: Gesamtbewertung für Option_logische_Kanäle

Prüfpunkt	Use Case	Kapitel	Bezeichnung	T_{Ri} / [ms]	g_i	g_i T_{Ri} / [ms]
~~GET RANDOM~~	~~(N099.322)~~	~~18.8.22~~	~~$P_{Random,b}$~~	~~4~~	~~1.800~~	~~7.200~~
			~~$P_{Random,m}$~~	~~40~~	200	~~8.000~~
MANAGE CHANNEL open	(N099.508)	18.8.23	P_{Open}	5	50.000	250.000
MANAGE CHANNEL close	(N099.514)		P_{Close}	3	40.000	120.000
MANAGE CHANNEL reset ICC	(N099.532)		P_{RST}	3	10.000	30.000
Spaltensummen				11	100.000	400.000

[<=]

A_16983 - (N109.480) K_Performanztest

Der Performanztest MUSS bei der Bewertung eines Prüflings die in CosT_439 dargestellten Werte für die Zulassungsgrenze zugrundelegen.

Tabelle 353: CosT_439: Gesamtbewertung je nach Kombination der Optionen

Basis / [ms]	Dual / [ms]	Krypto / [ms]	Kanal / [ms]	Summe / [s]	Zulassungsgrenze / [s]
12.687.572	375.000	1.875.000	400.000	15.337,572	6.710
12.687.572	375.000	1.875.000	0	14.937,572	6.535
12.687.572	375.000	0	400.000	13.462,672	5.890
12.687.572	375.000	0	0	13.062,572	5.715
12.687.572	0	1.875.000	400.000	14.962,572	6.546
12.687.572	0	1.875.000	0	14.562,572	6.371
12.687.572	0	0	400.000	13.087.672	5.726
12.687.572	0	0	0	12.687,572	5.551

[<=]

Hinweis CosH_d18: Bedeutung der Spaltenüberschriften in CosT_439:
- *a. Basis: COS, welches keines der optionalen Funktionspakete enthält*
- *b. Dual: COS enthält Option_kontaktlose_Schnittstelle*
- *c. Krypto: COS enthält Option_Kryptobox*
- *d. Kanal: COS enthält Option_logische_Kanäle*

A_16984 - (N109.500) K_Performanztest
Der Performanztest MUSS für die Gesamtbewertung folgende Formel verwenden, wobei nur die Prüfpunkte zu berücksichtigen sind, die gemäß der vom COS unterstützten Optionen vorhanden sind.

$$P_{gesamt} = \sum_i g_i \cdot P_i = \sum_i g_i \cdot f_{2i} \cdot T_{Ri}$$

[<=]

Unter den vereinfachenden Annahmen, dass die Messwerte innerhalb einer Reihe identisch sind (Standardabweichung ist null) und das Verhältnis X_i / T_{Ri} in allen Messreihen konstant ist, folgt dann für das Basisbetriebssystem, mit f_2 als Funktion von X_i / T_{Ri}.

[redacted]

CosA_3de zeigt diesen Zusammenhang graphisch mit dem Verhältnis X_i / T_{Ri} als Abszisse und $P_{gesamtEinfach}$ als Ordinate. Demnach erreicht ein unendlich schnelles Basisbetriebssystem 12.687,572s Punkte. Eine Karte, die in allen Prüfpunkten die Referenzzeit benötigt, erreicht 9515,679s Punkte. Eine Karte, die überall die doppelte Referenzzeit benötigt, null Punkte.

A_16985 - (N109.600) K_Performanztest
Der Performanztest MUSS es als zulassungsverhindernd werten, falls ein Prüfling in wenigstens einem Prüfpunkt das Verhältnis X_i / T_{Ri} größer als vier ist.**[<=]**

A_16986 - (N109.700) K_Performanztest
Der Performanztest MUSS es als zulassungsverhindernd werten, falls für einen Prüfling die gemäß (N109.500) ermittelte Punktzahl kleiner ist als in CosT_439 für die Kombination von Optionen angegebene. Dies entspricht einer relativen Ausführungszeit, die 1,5-mal so groß ist wie die Referenzzeit.**[<=]**

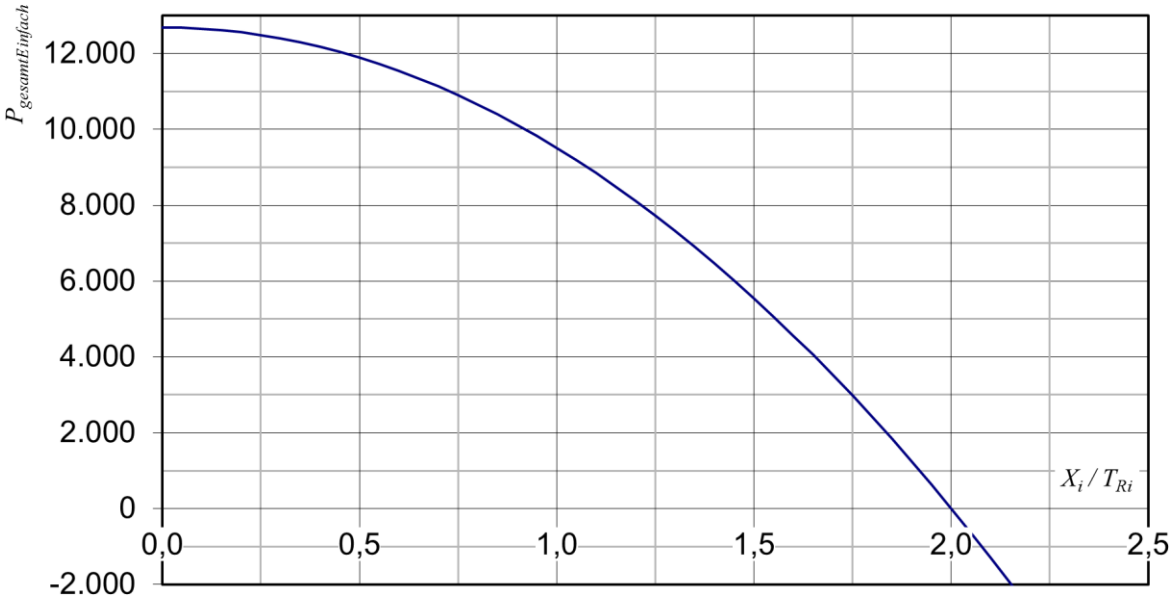


Abbildung 12: CosA_3de: Graphische Darstellung von $P_{gesamtEinfach}$

18.6 Übertragungsgeschwindigkeit

18.6.1 Übertragungsgeschwindigkeit für das Übertragungsprotokoll T=1

Dieser Prüfpunkt berücksichtigt die Kanalkapazität des Kanals TPDU_Channel für das Übertragungsprotokoll T=1 in CosA_a56. Diese wird wegen der starken Abhängigkeit von der externen Taktfrequenz des IFD nicht experimentell, sondern rechnerisch aus den Bytes TA_1 und TC_1 im ATR und der Kapazität C aus CosT_719 ermittelt.

A_16987 - (N200.100) K_Performanztest
Der Performanztest MUSS die Performanzpunkte P_{IO} wie folgt ermitteln:
Für die (rechnerische) Übertragungszeit von 1.000 Oktett zur Karte gilt:

Mit C aus CosT_719 für den Fall PPS1 = TA_1 und CGT in Abhängigkeit von TC_1 im ATR gemäß CosT_4ba:

Tabelle 354: CosT_4ba: Character Guard Time (CGT) gemäß [ISO/IEC 7816-3#11.2]

TC_1	'FF'	'00'	'01'	'02'	…	'FD'	'FE'
CGT	11	12	13	14	…	265	266

P_{IO} = points((t_T, t_T), T_{IO}).**[<=]**

18.6.2 Übertragungsgeschwindigkeit für das Protokoll [ISO/IEC 7816-12]

Hinweis CosH_75c: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

18.6.3 Übertragungsgeschwindigkeit für kontaktlose Datenübertragung

Hinweis CosH_025: Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

Die folgenden Punkte sind absichtlich leer:
(N109.800), (N109.900), (N110.000), (N110.100), (N110.200), (N110.300), (N110.400), (N110.500), (N110.600), (N110.700), (N110.800), (N110.900), (N111.000), (N111.100), (N111.200), (N111.300), (N111.400), (N111.500), (N111.600), (N111.700), (N111.800), (N111.900), (N112.000), (N112.100), (N112.200), (N112.300), (N112.400), (N112.500), (N112.600), (N112.700), (N112.800), (N112.900), (N113.000), (N113.100), (N113.200), (N113.300), (N113.400), (N113.500), (N113.600), (N113.700), (N113.800), (N113.900), (N114.000), (N114.100), (N114.200), (N114.300), (N114.400), (N114.500), (N114.600), (N114.700), (N114.800), (N114.900), (N115.000), (N115.100), (N115.200), (N115.300), (N115.400), (N115.500), (N115.600), (N115.700), (N115.800), (N115.900), (N116.000), (N116.100), (N116.200), (N116.300), (N116.400), (N116.500), (N116.600), (N116.700), (N116.800), (N116.900), (N117.000), (N117.100), (N117.200), (N117.300), (N117.400), (N117.500), (N117.600), (N117.700), (N117.800), (N117.900), (N118.000), (N118.100), (N118.200), (N118.300), (N118.400), (N118.500), (N118.600), (N118.700), (N118.800), (N118.900), (N119.000), (N119.100), (N119.200), (N119.300), (N119.400), (N119.500), (N119.600), (N119.700), (N119.800), (N119.900), (N120.000), (N120.100), (N120.200), (N120.300), (N120.400), (N120.500), (N120.600), (N120.700), (N120.800), (N120.900), (N121.000), (N121.100), (N122.000), (N122.100), (N122.200), (N122.300), (N122.400), (N122.500), (N122.600), (N122.700), (N122.800), (N122.900), (N123.000), (N123.100), (N123.200), (N123.300), (N123.400), (N123.500), (N123.600), (N123.700), (N123.800), (N123.900), (N124.000), (N124.100), (N124.200), (N124.300), (N124.400), (N124.500), (N124.600), (N124.700), (N124.710), (N124.800), (N124.900), (N125.000), (N125.100), (N125.200), (N125.300), (N125.400), (N125.500), (N125.600), (N125.700), (N125.800), (N125.900), (N126.000), (N126.100), (N126.200), (N126.300), (N126.400), (N126.500), (N126.600), (N126.700), (N126.800), (N126.900)

18.7 Startsequenz für das Übertragungsprotokoll T=1

Dieser Prüfpunkt behandelt eine Sequenz, die bei der Aktivierung einer Smartcard durchlaufen wird.

Der Prüfpunkt beinhaltet:

- Die Aktivierung (Bootvorgang) des Betriebssystems,
- die Aushandlung einer höheren Übertragungsrate und
- die Aushandlung einer Puffergröße für die Datenübertragung.

Testvorbereitung:

A_16988 - (N200.190) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

A_16989 - (N200.195) K_Performanztest
Schritt 2: Der Prüfling MUSS gemäß (N023.920)f deaktiviert werden.**[<=]**

~~Keine.~~

Testdurchführung:

G2_N200.200 - (N200.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt. Abweichend von den Festlegungen in 18.5.1.1 wird hier die Zeit pro Schleifendurchlauf anders ermittelt.**[<=]**

G2_N200.210 - (N200.210) K_Performanztest
Schritt 1: Die Zeit t_{Start} gibt den Zeitpunkt an, zu welchem die Smartcard gemäß (N108.800) aktiviert wird. Genauer, den Zeitpunkt des Wechsels von RST von L nach H (siehe [ISO/IEC 7816-3#Figure 1]). Anschließend wird der ATR empfangen.**[<=]**

G2_N200.220 - (N200.220) K_Performanztest
Schritt 2: Es MUSS eine PPS-Sequenz gemäß [ISO/IEC 7816-3] erfolgen. Die Bits 4 bis 1 in PPS0 MÜSSEN das Übertragungsprotokoll T=1 anzeigen. Als PPS1 MUSS der Wert von TA_1 aus dem ATR verwendet werden.**[<=]**

G2_N200.230 - (N200.230) K_Performanztest
Schritt 3: Das IFD MUSS der Smartcard den Wert von IFSD = 254 präsentieren.**[<=]**

G2_N200.240 - (N200.240) K_Performanztest
Schritt 4: Der Zeitpunkt t_{End} ist definiert durch das Ende der letzten TPDU, welche in (N200.230) übertragen wird. Damit gilt für den *i*-ten Schleifendurchlauf:
$t_{Run4,i} = t_{End} - t_{Start}$. **[<=]**

Testauswertung:

A_16990 - (N200.250) K_Performanztest
Der Performanztest MUSS die Performanzpunkte $P_{KarteStarten}$ wie folgt ermitteln:
$P_{KarteStarten}$ = points(($t_{Run4,1}$, $t_{Run4,2}$, …, $t_{Run4,100}$), $T_{KarteStarten}$).**[<=]**

Testnachbereitung:

Keine.

Hinweis CosH_98e: Besser wäre es, wenn auch hier die Messung aus 18.5.1.1 anwendbar wäre. Es ist zu prüfen, ob dafür passendes Equipment zur Verfügung steht.

18.8 Messverfahren für Einzelkommandos (normativ)

18.8.1 ACTIVATE, DEACTIVATE, DELETE, LOAD APPLICATION, TERMINATE

In diesem Kapitel werden Kommandos zur Bearbeitung eines Life Cycle Status betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.LCS.

Testvorbereitung:

G2_N201.110 - (N201.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N201.120 - (N201.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.LCS gesetzt werden.**[<=]**

Testdurchführung:

G2_N201.200 - (N201.200) K_Performanztest

Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 10 ausgeführt.**[<=]**

G2_N201.210 - (N201.210) K_Performanztest
Schritt 1: Die Datei / MF / DF.LCS / EF.LCS MUSS mittels Use Case aus (N046.700) selektiert werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

Schritt 2: LCS für Dateien

G2_N201.220.a - (N201.220)a K_Performanztest
Schritt 2a: *currentEF* MUSS mittels Use Case aus (N036.000) deaktiviert werden. Die Laufzeit $t_{deactivate_EF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.220.b - (N201.220)b K_Performanztest
Schritt 2b: *currentEF* MUSS mittels Use Case aus (N034.800) aktiviert werden. Die Laufzeit $t_{activate_EF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.220.c - (N201.220)c K_Performanztest
Schritt 2c: *currentEF* MUSS mittels Use Case aus (N048.903) terminiert werden. Die Laufzeit $t_{terminate_EF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.220.d - (N201.220)d K_Performanztest
Schritt 2d: *currentEF* MUSS mittels Use Case aus (N037.100) gelöscht werden. Die Laufzeit $t_{delete_EF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Schritt 3: Arbeiten mit Passwortobjekten:

G2_N201.230.a - (N201.230)a K_Performanztest
Schritt 3a: Der Status von PIN.LCS MUSS mittels Use Case aus (N077.990) abgefragt werden. Die Laufzeit $t_{GetStatus,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.b - (N201.230)b K_Performanztest
Schritt 3b: Der Transportschutz von PIN.LCS MUSS mittels Use Case aus (N073.700) aufgehoben werden, wobei für *newSecret* eine zufällig gewählte PIN verwendet werden MUSS, deren Länge gleichverteilt aus dem Bereich [6, 12] gezogen wird. Die Laufzeit $t_{SetPIN,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.c - (N201.230)c K_Performanztest
Schritt 3c: Das Attribut PIN.LCS.*flagEnabled* MUSS mittels Use Case aus (N075.386) auf den Wert False gesetzt werden. Die Laufzeit $t_{DisablePIN,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.d - (N201.230)d K_Performanztest
Schritt 3d: Das Attribut PIN.LCS.*flagEnabled* MUSS mittels Use Case aus (N076.586) auf den Wert True gesetzt werden. Die Laufzeit $t_{EnablePIN,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.e - (N201.230)e K_Performanztest
Schritt 3e: Das Attribut PIN.LCS.*secret* MUSS mittels Use Case aus (N073.300) auf einen anderen Wert gesetzt werden, wobei für *newSecret* eine zufällig gewählte PIN verwendet werden MUSS, deren Länge gleichverteilt aus dem Bereich [6, 12] gezogen

wird. Die Laufzeit $t_{ChRefData,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.f - (N201.230)f K_Performanztest
Schritt 3f: VERIFY Kommando gemäß (N082.200) mit korrekten *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{VERIFY_1,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.g - (N201.230)g K_Performanztest
Schritt 3g: VERIFY Kommando gemäß (N082.200) mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{VERIFY_2,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.h - (N201.230)h K_Performanztest
Schritt 3h: VERIFY Kommando gemäß (N082.200) mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{VERIFY_3,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.230.i - (N201.230)i K_Performanztest
Schritt 3i: VERIFY Kommando gemäß (N082.200) mit korrekten *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{VERIFY_4,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

A_16991 - (N201.230)j K_Performanztest
Schritt 3j: VERIFY Kommando gemäß (N082.200) mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N201.230.k - (N201.230)k K_Performanztest
Schritt 3k: RESET RETRY COUNTER gemäß (N079.300) mit korrekter PUK für das Objekt PIN.LCS, wobei für *newSecret* eine zufällig gewählte PIN verwendet werden MUSS, deren Länge gleichverteilt aus dem Bereich [6, 12] gezogen wird. Die Laufzeit $t_{ResetRC,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Schritt 4: LCS für Passwortobjekte:

G2_N201.240.a - (N201.240)a K_Performanztest
Schritt 4a: PIN.LCS MUSS mittels Use Case aus (N036.034) deaktiviert werden. Die Laufzeit $t_{deactivate_Pwd,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.240.b - (N201.240)b K_Performanztest
Schritt 4b: PIN.LCS MUSS mittels Use Case aus (N034.834) aktiviert werden. Die Laufzeit $t_{activate_Pwd,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.240.c - (N201.240)c K_Performanztest
Schritt 4c: PIN.LCS MUSS mittels Use Case aus (N048.934) terminiert werden. Die Laufzeit $t_{terminate_Pwd,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.240.d - (N201.240)d K_Performanztest
Schritt 4d: PIN.LCS MUSS mittels Use Case aus (N037.134) gelöscht werden. Die Laufzeit $t_{delete_Pwd,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Schritt 5: LCS für private Schlüssel:

G2_N201.250.a - (N201.250)a K_Performanztest
Schritt 5a: PrK.LCS MUSS mittels Use Case aus (N036.014) deaktiviert werden. Die Laufzeit $t_{deactivate_PrK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.250.b - (N201.250)b K_Performanztest
Schritt 5b: PrK.LCS MUSS mittels Use Case aus (N034.814) aktiviert werden. Die Laufzeit $t_{activate_PrK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.250.c - (N201.250)c K_Performanztest
Schritt 5c: PrK.LCS MUSS mittels Use Case aus (N048.914) terminiert werden. Die Laufzeit $t_{terminate_PrK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.250.d - (N201.250)d K_Performanztest
Schritt 5d: PrK.LCS MUSS mittels Use Case aus (N037.114) gelöscht werden. Die Laufzeit $t_{delete_PrK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Schritt 6: LCS für öffentliche Schlüssel:

G2_N201.260.a - (N201.260)a K_Performanztest
Schritt 6a: PuK.LCS MUSS mittels Use Case aus (N036.024) deaktiviert werden. Die Laufzeit $t_{deactivate_PuK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.260.b - (N201.260)b K_Performanztest
Schritt 6b: PuK.LCS MUSS mittels Use Case aus (N034.824) aktiviert werden. Die Laufzeit $t_{activate_PuK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.260.c - (N201.260)c K_Performanztest
Schritt 6c: PuK.LCS MUSS mittels Use Case aus (N048.924) terminiert werden. Die Laufzeit $t_{terminate_PuK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.260.d - (N201.260)d K_Performanztest
Schritt 6d: PuK.LCS MUSS mittels Use Case aus (N037.124) gelöscht werden. Die Laufzeit $t_{delete_PuK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Schritt 7: LCS für symmetrische Schlüssel:

G2_N201.270.a - (N201.270)a K_Performanztest
Schritt 7a: SK.LCS MUSS mittels Use Case aus (N036.014) deaktiviert werden. Die Laufzeit $t_{deactivate_SyK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.270.b - (N201.270)b K_Performanztest
Schritt 7b: SK.LCS MUSS mittels Use Case aus (N034.814) aktiviert werden. Die Laufzeit $t_{activate_SyK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.270.c - (N201.270)c K_Performanztest

Schritt 7c: SK.LCS MUSS mittels Use Case aus (N048.914) terminiert werden. Die Laufzeit $t_{terminate_SyK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.270.d - (N201.270)d K_Performanztest
Schritt 7d: SK.LCS MUSS mittels Use Case aus (N037.114) gelöscht werden. Die Laufzeit $t_{delete_SyK,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.280 - (N201.280) K_Performanztest
Schritt 8: Der Ordner / MF / DF.LCS MUSS mittels Use Case aus (N044.900) selektiert werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

Schritt 9: LCS für Ordner:

G2_N201.290.a - (N201.290)a K_Performanztest
Schritt 9a: *currentFolder* MUSS mittels Use Case aus (N036.000) deaktiviert werden. Die Laufzeit $t_{deactivate_DF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.290.b - (N201.290)b K_Performanztest
Schritt 9b: *currentFolder* MUSS mittels Use Case aus (N034.800) aktiviert werden. Die Laufzeit $t_{activate_DF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.290.c - (N201.290)c K_Performanztest
Schritt 9c: *currentFolder* MUSS mittels Use Case aus (N048.800) terminiert werden. Die Laufzeit $t_{terminate_DF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N201.290.d - (N201.290)d K_Performanztest
Schritt 9d: *currentFolder* MUSS mittels Use Case aus (N037.100) gelöscht werden. Die Laufzeit $t_{delete_DF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Schritt 10: Laden von DF.LCS

G2_N201.300.a - (N201.300)a K_Performanztest
Schritt 10a: Die Anwendung DF.LCS MUSS mittels LOAD APPLICATION in den Prüfling geladen werden. Dazu sind ein oder mehrere LOAD APPLICATION Kommandos erforderlich. Gemäß (N040.100) ist *currentFolder* anschließend auf DF.LCS gesetzt.**[<=]**

G2_N201.300.b - (N201.300)b K_Performanztest
Schritt 10b: Die Ausführungszeit jedes LOAD APPLICATION Kommandos MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

A_16992 - (N201.300)c K_Performanztest
Die Summe aller LOAD APPLICATION Kommandoausführungszeiten in der *i*-ten Schleifeniteration wird mit $t_{LoadApp,i}$ bezeichnet. **[<=]**

Testauswertung:

A_16993 - (N201.310) K_Performanztest
Der Performanztest MUSS die in CosT_3d5 genannten Performanzpunkte wie dort angegeben ermitteln:

Tabelle 355: CosT_3d5: Performanzpunkte für LCS Operationen

$P_{activate_DF}$	= points(($t_{activate_DF,1}$,	$t_{activate_DF,2}$, …,	$t_{activate_DF,100}$),	$T_{activate_DF}$)
$P_{activate_EF}$	= points(($t_{activate_EF,1}$,	$t_{activate_EF,2}$, …,	$t_{activate_EF,100}$),	$T_{activate_EF}$)
$P_{activate_Pwd}$	= points(($t_{activate_Pwd,1}$,	$t_{activate_Pwd,2}$, …,	$t_{activate_Pwd,100}$),	$T_{activate_Pwd}$)
$P_{activate_PrK}$	= points(($t_{activate_PrK,1}$,	$t_{activate_PrK,2}$, …,	$t_{activate_PrK,100}$),	$T_{activate_PrK}$)
$P_{activate_PuK}$	= points(($t_{activate_PuK,1}$,	$t_{activate_PuK,2}$, …,	$t_{activate_PuK,100}$),	$T_{activate_PuK}$)
$P_{activate_SK}$	= points(($t_{activate_SK,1}$,	$t_{activate_SK,2}$, …,	$t_{activate_SK,100}$),	$T_{activate_SK}$)
$P_{deactivate_DF}$	= points(($t_{deactivate_DF,1}$,	$t_{deactivate_DF,2}$, …,	$t_{deactivate_DF,100}$),	$T_{deactivate_DF}$)
$P_{deactivate_EF}$	= points(($t_{deactivate_EF,1}$,	$t_{deactivate_EF,2}$, …,	$t_{deactivate_EF,100}$),	$T_{deactivate_EF}$)
$P_{deactivate_Pwd}$	= points(($t_{deactivate_Pwd,1}$,	$t_{deactivate_Pwd,2}$, …,	$t_{deactivate_Pwd,100}$),	$T_{deactivate_Pwd}$)
$P_{deactivate_PrK}$	= points(($t_{deactivate_PrK,1}$,	$t_{deactivate_PrK,2}$, …,	$t_{deactivate_PrK,100}$),	$T_{deactivate_PrK}$)
$P_{deactivate_PuK}$	= points(($t_{deactivate_PuK,1}$,	$t_{deactivate_PuK,2}$, …,	$t_{deactivate_PuK,100}$),	$T_{deactivate_PuK}$)
$P_{deactivate_SK}$	= points(($t_{deactivate_SK,1}$,	$t_{deactivate_SK,2}$, …,	$t_{deactivate_SK,100}$),	$T_{deactivate_SK}$)
P_{delete_DF}	= points(($t_{delete_DF,1}$,	$t_{delete_DF,2}$, …,	$t_{delete_DF,100}$),	T_{delete_DF})
P_{delete_EF}	= points(($t_{delete_EF,1}$,	$t_{delete_EF,2}$, …,	$t_{delete_EF,100}$),	T_{delete_EF})
P_{delete_Pwd}	= points(($t_{delete_Pwd,1}$,	$t_{delete_Pwd,2}$, …,	$t_{delete_Pwd,100}$),	T_{delete_Pwd})
P_{delete_PrK}	= points(($t_{delete_PrK,1}$,	$t_{delete_PrK,2}$, …,	$t_{delete_PrK,100}$),	T_{delete_PrK})
P_{delete_PuK}	= points(($t_{delete_PuK,1}$,	$t_{delete_PuK,2}$, …,	$t_{delete_PuK,100}$),	T_{delete_PuK})
P_{delete_SK}	= points(($t_{delete_SK,1}$,	$t_{delete_SK,2}$, …,	$t_{delete_SK,100}$),	T_{delete_SK})
$P_{terminate_DF}$	= points(($t_{terminate_DF,1}$,	$t_{terminate_DF,2}$, …,	$t_{terminate_DF,100}$),	$T_{terminate_DF}$)
$P_{terminate_EF}$	= points(($t_{terminate_EF,1}$,	$t_{terminate_EF,2}$, …,	$t_{terminate_EF,100}$),	$T_{terminate_EF}$)
$P_{terminate_Pwd}$	= points(($t_{terminate_Pwd,1}$,	$t_{terminate_Pwd,2}$, …,	$t_{terminate_Pwd,100}$),	$T_{terminate_Pwd}$)
$P_{terminate_PrK}$	= points(($t_{terminate_PrK,1}$,	$t_{terminate_PrK,2}$, …,	$t_{terminate_PrK,100}$),	$T_{terminate_PrK}$)
$P_{terminate_PuK}$	= points(($t_{terminate_PuK,1}$,	$t_{terminate_PuK,2}$, …,	$t_{terminate_PuK,100}$),	$T_{terminate_PuK}$)
$P_{terminate_SK}$	= points(($t_{terminate_SK,1}$,	$t_{terminate_SK,2}$, …,	$t_{terminate_SK,100}$),	$T_{terminate_SK}$)
$P_{LoadApp}$	= points(($t_{LoadApp,1}$,	$t_{LoadApp,2}$, …,	$t_{LoadApp,100}$),	$T_{LoadApp}$)
$P_{GetPinStatus}$	= points(($t_{GetStatus,1}$,	$t_{GetStatus,2}$, …,	$t_{GetStatus,100}$),	$T_{GetStatus}$)
P_{SetPIN}	= points(($t_{SetPIN,1}$,	$t_{SetPIN,2}$, …,	$t_{SetPIN,100}$),	T_{SetPIN})
$P_{DisablePIN}$	= points(($t_{DisablePIN,1}$,	$t_{DisablePIN,2}$, …,	$t_{DisablePIN,100}$),	$T_{DisablePIN}$)
$P_{EnablePIN}$	= points(($t_{EnablePIN,1}$,	$t_{EnablePIN,2}$, …,	$t_{EnablePIN,100}$),	$T_{EnablePIN}$)
$P_{ChRefData}$	= points(($t_{ChRefData,1}$,	$t_{ChRefData,2}$, …,	$t_{ChRefData,100}$),	$T_{ChRefData}$)
P_{VERIFY_1}	= points(($t_{VERIFY_1,1}$,	$t_{VERIFY_1,2}$, …,	$t_{VERIFY_1,100}$),	0,8 T_{VERIFY})

P_{VERIFY_2} = points(($t_{VERIFY_2,1}$, $t_{VERIFY_2,2}$, …, $t_{VERIFY_2,100}$), T_{VERIFY})

P_{VERIFY_3} = points(($t_{VERIFY_3,1}$, $t_{VERIFY_3,2}$, …, $t_{VERIFY_3,100}$), T_{VERIFY})

P_{VERIFY_4} = points(($t_{VERIFY_4,1}$, $t_{VERIFY_4,2}$, …, $t_{VERIFY_4,100}$), T_{VERIFY})

P_{VERIFY} = 0,8 P_{VERIFY_1} + 0,08 P_{VERIFY_2} + 0,02 P_{VERIFY_3} + 0,1 P_{VERIFY_4}

$P_{ResetRC}$ = points(($t_{ResetRC,1}$, $t_{ResetRC,2}$, …, $t_{ResetRC,100}$), $T_{ResetRC}$)

[<=]

Testnachbereitung:

Keine.

18.8.2 SELECT Datei

In diesem Kapitel wird die Selektion von Dateien betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.SelectEF.

Testvorbereitung:

G2_N202.110 - (N202.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N202.120 - (N202.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.SelectEF gesetzt werden.**[<=]**

G2_N202.130 - (N202.130) K_Performanztest
Schritt 3: *currentEF* MUSS auf / MF / DF.SelectEF / EF.00 gesetzt werden.**[<=]**

Testdurchführung:

G2_N202.200 - (N202.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.**[<=]**

A_16994 - (N202.210) K_Performanztest
Schritt 1: Aus der Menge der zu Anfang 100 in DF.SelectEF gültigen Fileidentifier MUSS ein bislang noch nicht verwendeter gezogen werden (ziehen ohne zurücklegen).**[<=]**

G2_N202.220 - (N202.220) K_Performanztest
Schritt 2: Die Datei mit dem in Schritt 1 ausgewählten Fileidentifier MUSS mittels Use Case aus (N046.700) selektiert werden. Die Laufzeit $t_{select_EF,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_16995 - (N202.230) K_Performanztest
Der Performanztest MUSS die Performanzpunkte P_{select_EF} wie folgt ermitteln:
P_{select_EF} = points($t_{select_EF,1}$, $t_{select_EF,2}$, …, $t_{select_EF,100}$), T_{select_EF}) **[<=]**

Testnachbereitung:

Keine.

18.8.3 FINGERPRINT

In diesem Abschnitt wird die Berechnung des COS Fingerprints betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF.

Testvorbereitung:

G2_N203.110 - (N203.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

Testdurchführung:

A_16996 - (N203.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.**[<=]**

G2_N203.210 - (N203.210) K_Performanztest
Schritt 1: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *prefix* = RAND(128) erzeugt werden.**[<=]**

G2_N203.220 - (N203.220) K_Performanztest
Schritt 2: FINGERPRINT Kommando gemäß (N096.454), wobei als *prefix* der im vorherigen Schritt erzeugte Wert dient. Die Laufzeit $t_{fp,i}$ dieses Kommandos MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_16997 - (N203.230) K_Performanztest
Der Performanztest MUSS die Performanzpunkte $P_{fingerprint}$ wie folgt ermitteln:
$P_{fingerprint}$ = points(($t_{fp,1}$, $t_{fp,2}$, …, $t_{fp,100}$), $T_{fingerprint}$) **[<=]**

Testnachbereitung:

Keine.

18.8.4 TERMINATE CARD USAGE

In diesem Kapitel wird das Terminieren einer Smartcard betrachtet.

Testvorbereitung:

G2_N204.110 - (N204.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

Testdurchführung:

G2_N204.200 - (N204.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife zehnmal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.**[<=]**

G2_N204.210 - (N204.210) K_Performanztest
Schritt 1: Die Smartcard MUSS mittels Use Case aus (N048.700) terminiert werden. Die Laufzeit $t_{terminateCard,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N204.220 - (N204.220) K_Performanztest
Schritt 2: Die Smartcard MUSS so reinitialisiert werden, dass die in 18.9 definierte Kartenkonfiguration vorliegt. Die Ausführungszeit dieser Operation ist für diesen Prüfpunkt irrelevant.**[<=]**

Testauswertung:

A_16998 - (N204.230) K_Performanztest
Der Performanztest MUSS die Performanzpunkte $P_{terminateCard}$ wie folgt ermitteln:
$P_{terminateCard}$ = points(($t_{terminateCard,1}$, $t_{terminateCard,2}$,…, $t_{terminateCard,10}$), $T_{terminateCard}$) **[<=]**

Testnachbereitung:

Keine.

18.8.5 SET LOGICAL EOF, WRITE BINARY

In diesem Kapitel wird das Anfügen von Daten in transparenten EF und das Setzen des Attributes *positionLogicalEndOfFile* betrachtet. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.SelectEF / EF.transparent.

Testvorbereitung:

G2_N205.110 - (N205.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N205.120 - (N205.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.**[<=]**

G2_N205.130 - (N205.130) K_Performanztest
Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.**[<=]**

G2_N205.140 - (N205.140) K_Performanztest
Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen M_{Nc} = {1, 2, 3, …, 1000} erstellt werden.**[<=]**

G2_N205.150 - (N205.150) K_Performanztest
Schritt 5: Es MUSS eine leere Menge M_{SetEOF} = {} erstellt werden.**[<=]**

Testdurchführung:

A_17000 - (N205.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife solange durchlaufen, bis die Menge M_{Nc} leer ist. In jedem Schleifendurchlauf werden die Schritte 1 bis 3 ausgeführt. Nach Abarbeitung der Schleife wird mit Schritt 4 fortgefahren.**[<=]**

A_17001 - (N205.210) K_Performanztest
Schritt 1: Aus der Menge M_{Nc} wird ein beliebiges Element *e* zufällig gezogen (ziehen ohne zurücklegen) und der Oktettstring *newData* = RAND(*e*) erzeugt.**[<=]**

G2_N205.220 - (N205.220) K_Performanztest
Schritt 2: Der *body* von EF.transparent wird gemäß Use Case aus (N055.205) erweitert wobei *newData* als Kommandonachricht verwendet wird. Die Laufzeit $t_{write,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N205.230 - (N205.230) K_Performanztest
Schritt 3: Wenn das Attribute EF.transparent.*positionLogicalEndOfFile* nach Schritt 2 größer als '7800' = 30.720 ist, dann MUSS dieses Attribut mittels Use Case aus (N052.932) auf den Wert null gesetzt werden. Die Laufzeitt_{set} dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und der Menge M_{SetEOF} hinzugefügt werden.**[<=]**

G2_N205.240 - (N205.240) K_Performanztest
Schritt 4: Das Attribute EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus (N052.932) auf den Wert null gesetzt. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt

irrelevant.~~Die Laufzeit t_{set} dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und der Menge M_{SetEOF} hinzugefügt werden.~~**[<=]**

Testauswertung:

A_17002 - (N205.250) Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:
Durch die Messpunkt (x, y) → (e, $t_{write,i}$) wird eine Ausgleichsgerade $y = m\,x + b$ gelegt.

$$P_{wirte,b} = \text{points}(b, b) , T_{write,b})$$

$$P_{wirte,m} = \text{points}(1000\,m, 1000\,m) , T_{wirte,b})$$

$$P_{SetEOF} = \text{points}(M_{SetEOF} , T_{SetEOF})$$

[<=]

Hinweis CosH_375: Die Steigung m *der Ausgleichsgeraden gibt die Schreibrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von* P_m *aufgeführt ist, rechnet dies um in Sekunden pro kByte.*

Testnachbereitung:

Keine.

18.8.6 ERASE BINARY, UPDATE BINARY

In diesem Kapitel wird das Schreiben und Löschen von Daten in transparenten EF betrachtet. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.SelectEF / EF.transparent.

Testvorbereitung:

G2_N206.110 - (N206.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N206.120 - (N206.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.**[<=]**

G2_N206.130 - (N206.130) K_Performanztest
Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.**[<=]**

G2_N206.140 - (N206.140) K_Performanztest
Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen M_{Nc} = {1, 2, 3, …, 1000} erstellt werden.**[<=]**

G2_N206.150 - (N206.150) K_Performanztest
Schritt 5: Es MUSS eine leere Menge M_{wipe} = {} erstellt werden.**[<=]**

A_17003 - (N206.160) K_Performanztest
Schritt 6: Das Attribute EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus (N053.200) auf den Wert EF.transparent.*numberOfOctet* gesetzt, wobei für den Kommandoparameter *newData* = '00' gilt.**[<=]**

A_17004 - (N206.170) K_Performanztest
Schritt 7: Die Variable *index* wird auf den Wert null gesetzt.**[<=]**

Testdurchführung:

A_17005 - (N206.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife solange, bis die Menge M_{Nc} leer ist. In jedem Schleifendurchlauf werden die Schritte 1 bis 3 ausgeführt. Nach Abarbeitung der Schleife wird mit Schritt 4 fortgefahren.**[<=]**

A_17006 - (N206.210) K_Performanztest
Schritt 1: Aus der Menge M_{Nc} wird ein beliebiges Element *e* zufällig gezogen (ziehen ohne zurücklegen) und der Oktettstring *newData* = RAND(*e*) erzeugt.**[<=]**

G2_N206.220 - (N206.220) K_Performanztest
Schritt 2: Der *body* von EF.transparent wird gemäß Use Case aus (N053.200) beschrieben, wobei *newData* als Kommandonachricht und *index* als Kommandoparameter *offset* verwendet wird. Die Laufzeit $t_{update,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden. Nach Kommandoausführung wird *index* += *e* gesetzt. **[<=]**

G2_N206.230 - (N206.230) K_Performanztest
Schritt 3: Wenn *index* nach Schritt 2 größer als '7800' = 30.720 ist, dann MUSS *index* auf den Wert null gesetzt werden und der Inhalt von EF.transparent mittels Use Case aus (N049.100) gelöscht werden, wobei der Kommandoparameter *offset* = 0 gesetzt wird. Die Laufzeit t_{wipe} dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und der Menge M_{wipe} hinzugefügt werden.**[<=]**

G2_N206.240 - (N206.240) K_Performanztest
Schritt 4: Der Inhalt von EF.transparent wird mittels Use Case aus (N049.100) gelöscht, wobei der Kommandoparameter *offset* = 0 gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.~~Die Laufzeit t_{wipe} dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und der Menge M_{wipe} hinzugefügt werden.~~**[<=]**

Testauswertung:

A_17007 - (N206.250) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:
Durch die Messpunkt (*x*, *y*) → (e, $t_{update,i}$) wird eine Ausgleichsgerade *y* = *m x* + *b* gelegt.

$P_{UpdateBin,b}$ = points((*b*, *b*) , $T_{update,b}$)

$P_{UpdateBin,m}$ = points((1000 *m*, 1000 *m*) , $T_{update,m}$)

$P_{WipeBin}$ = points(M_{wipe} , T_{wipe})

[<=]

Hinweis CosH_f74: Die Steigung m *der Ausgleichsgeraden gibt die Schreibrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von* P_m *aufgeführt ist, rechnet dies um in Sekunden pro kByte.*

Testnachbereitung:

A_17009 - (N206.300) K_Performanztest
Das Attribute EF.transparent.*positionLogicalEndOfFile* MUSS mittels Use Case aus (N052.932) auf den Wert null gesetzt werden.**[<=]**

18.8.7 READ BINARY

In diesem Kapitel wird das Lesen von Daten in transparenten EF betrachtet. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.SelectEF / EF.transparent.

Testvorbereitung:

G2_N207.110 - (N207.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N207.120 - (N207.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.**[<=]**

G2_N207.130 - (N207.130) K_Performanztest
Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.**[<=]**

G2_N207.140 - (N207.140) K_Performanztest
Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen M_{Ne}= {1, 2, 3, …, 1000} erstellt werden.**[<=]**

A_17010 - (N207.150) K_Performanztest
Schritt 5: Das Attribute EF.transparent.*body* MUSS vollständig mit zufälligen Werten befüllt werden.**[<=]**

Testdurchführung:

A_17011 - (N207.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife solange durchlaufen, bis die Menge M_{Ne} leer ist. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.**[<=]**

A_17012 - (N207.210) K_Performanztest
Schritt 1: Aus der Menge M_{Ne} wird ein beliebiges Element *e* zufällig gezogen (ziehen ohne zurücklegen).**[<=]**

G2_N207.220 - (N207.220) K_Performanztest
Schritt 2: Teile von EF.transparent.*body* werden gemäß Use Case aus (N051.100) ausgelesen, wobei *e* als Kommandoparameter *length* und eine zufällige Zahl aus dem Bereich [0, 30.720] = ['0000', '7800'] als Kommandoparameter*offset* verwendet wird. Die Laufzeit $t_{read,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17013 - (N207.230) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:
Durch die Messpunkt (x, y) → (e, $t_{read,i}$) wird eine Ausgleichsgerade $y = m\,x + b$ gelegt.

$P_{ReadBinary,b}$	= points(	(b, b)	,	$T_{ReadBinary,b}$)
$P_{ReadBinary,m}$	= points(	(1000 m, 1000 m)	,	$T_{ReadBinary,m}$)

[<=]

Hinweis CosH_8e2: Die Steigung m *der Ausgleichsgeraden gibt die Leserate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von* P_m *aufgeführt ist, rechnet dies um in Sekunden pro kByte.*

Testnachbereitung:

A_17014 - (N207.300) K_Performanztest
Das Attribute EF.transparent.*positionLogicalEndOfFile* MUSS mittels Use Case aus (N052.932) auf den Wert null gesetzt werden.**[<=]**

18.8.8 Rekord orientierte Kommandos

In diesem Abschnitt werden folgende Kommandos betrachtet: ACTIVATE RECORD, APPEND RECORD, DEACTIVATE RECORD, DELETE RECORD, ERASE RECORD, READ RECORD, UPDATE RECORD. Das Kommando SEARCH RECORD wird in 18.8.9 behandelt. Dieser Prüfpunkt arbeitet mit der Datei
/ MF / DF.strukturiert / EF.strukturiert.

Testvorbereitung:

G2_N208.100 - (N208.100) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N208.110 - (N208.110) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.strukturiert gesetzt werden.**[<=]**

G2_N208.120 - (N208.120) K_Performanztest
Schritt 3: Die Datei / MF / DF.strukturiert / EF.strukturiert MUSS mittels Use Case aus (N46.700) selektiert werden.**[<=]**

Testdurchführung:

G2_N208.200 - (N208.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 255 mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 7 ausgeführt.**[<=]**

G2_N208.210 - (N208.210) K_Performanztest
Schritt 1, APPEND RECORD: Im *i*-ten Schleifendurchlauf wird ein Oktettstring *recordData* = RAND(*i*) erzeugt. In *currentEF* MUSS mittels Use Case aus (N058.400) ein neuer Rekord angelegt werden, wobei als Datenteil der Kommandonachricht *recordData* verwendet wird. Die Laufzeit $t_{append,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N208.220 - (N208.220) K_Performanztest
Schritt 2, DEACTIVATE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus (N060.700) deaktiviert werden. Die Laufzeit $t_{deactivate,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N208.230 - (N208.230) K_Performanztest
Schritt 3, ACTIVATE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus (N055.500) aktiviert werden. Die Laufzeit $t_{activate,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N208.240 - (N208.240) K_Performanztest
Schritt 4, UPDATE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus (N070.300) auf den Wert *newData* = RAND(*i*) werden. Die Laufzeit $t_{update,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N208.250 - (N208.250) K_Performanztest
Schritt 5, READ RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus (N065.700) gelesen werden. Die Laufzeit $t_{read,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N208.260 - (N208.260) K_Performanztest
Schritt 6, ERASE RECORD wipe: Der Inhalt von Rekord 1 in *currentEF* MUSS mittels Use Case aus (N063.600) gelöscht werden. Die Laufzeit $t_{wipe,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N208.270 - (N208.270) K_Performanztest
Schritt 7, DELETE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus (N063.422) aus *currentEF. recordList* entfernt werden. Die Laufzeit $t_{delete,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17015 - (N208.280) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:
Durch die Messpunkt (x, y) → (i, $t_{a,i}$) wird eine Ausgleichsgerade $y = m_a\, x + b_a$ gelegt.

$P_{AppendRecord,b}$	= points(	($b_{AppendRecord}$,	$b_{AppendRecord}$),	$T_{AppendRecord,b}$)
$P_{AppendRecord,m}$	= points(	(1000 $m_{AppendRecord}$,	1000 $m_{AppendRecord}$),	$T_{AppendRecord,m}$)
$P_{UpdateRecord,b}$	= points(	($b_{UpdateRecord}$,	$b_{UpdateRecord}$),	$T_{UpdateRecord,b}$)
$P_{UpdateRecord,m}$	= points(	(1000 $m_{UpdateRecord}$,	1000 $m_{UpdateRecord}$),	$T_{UpdateRecord,m}$)
$P_{ReadRecord,b}$	= points(	($b_{ReadRecord}$,	$b_{ReadRecord}$),	$T_{ReadRecord,b}$)
$P_{ReadRecord,m}$	= points(	(1000 $m_{ReadRecord}$,	1000 $m_{ReadRecord}$),	$T_{ReadRecord,m}$)
$P_{WipeRecord,b}$	= points(	($b_{WipeRecord}$,	$b_{WipeRecord}$),	$T_{WipeRecord,b}$)
$P_{WipeRecord,m}$	= points(	(1000 $m_{WipeRecord}$,	1000 $m_{WipeRecord}$),	$T_{WipeRecord,m}$)
$P_{ActivateRec}$	= points(($t_{ActivateRec,1}$,	$t_{ActivateRec,2}$, …,	$t_{ActivateRec,1000}$),	$T_{ActivateRec}$)
$P_{DeactivateRec}$	= points(($t_{DeactivateRec,1}$,	$t_{DeactivateRec,2}$, …,	$t_{DeactivateRec,1000}$),	$T_{DeactivateRec}$)
$P_{DeleteRec}$	= points(($t_{DeleteRec,1}$,	$t_{DeleteRec,2}$, …,	$t_{DeleteRec,1000}$),	$T_{DeleteRec}$)

[<=]

Hinweis CosH_1c3: Die Steigung m *der Ausgleichsgeraden gibt die Schreib- oder Leserate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von* $P_{a,m}$ *aufgeführt ist, rechnet dies um in Sekunden pro kByte.*

Testnachbereitung:

Keine.

18.8.9 SEARCH RECORD

Hier wird lediglich das Kommando SEARCH RECORD betrachtet. Die übrigen rekordorientierten Kommandos werden in 18.8.8 behandelt. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.strukturiert / EF.strukturiert.

Testvorbereitung:

G2_N209.110 - (N209.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N209.120 - (N209.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.strukturiert gesetzt werden.**[<=]**

G2_N209.130 - (N209.130) K_Performanztest
Schritt 3: Die Datei / MF / DF.strukturiert / EF.strukturiert MUSS mittels Use Case aus (N046.700) selektiert werden.**[<=]**

G2_N209.140 - (N209.140) K_Performanztest
Schritt 4: Das Attribut *currentEF. recordList* MUSS mittels Use Case aus (N058.400) wie folgt mit 254 Rekords gefüllt werden, die alle 255 Oktette lang sind:
a. Rekord 1 = '0100FF…FF',
b. Rekord 2 = 'FF0200FF…FF',
c. Rekord 3 = 'FFFF0300FF…FF',
d. …
e. Rekord 16 = 'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1000FF…FF',
f. ...
g. Rekord 252 = 'FF…FFFC00FFFF',
h. Rekord 253 = 'FF…FFFD00FF',
i. Rekord 254 = 'FF…FFFE00'.**[<=]**

Hinweis CosH_0f3: Ein Rekord i *enthält seine Rekordnummer* n *an der Position* n. *Der Rekordnummer folgt ein Oktett mit dem Wert '00'. Die übrigen Oktette besitzen den Wert 'FF'. Für ein Oktett lange Pattern gilt somit:*
a. Pattern = '00': Dieses Pattern ist in jedem Rekord enthalten. Der offset *dieses Patterns variiert mit der Rekordnummer.*
b. Pattern = 'FF': Dieses Pattern ist in jedem Rekord im Wesentlichen am Rekordanfang enthalten.
c. Alle übrigen Pattern mit einer Länge von einem Oktett sind in genau einem Rekord enthalten. Der offset *dieser Pattern variiert mit der Rekordnummer.*

Testdurchführung:

G2_N209.200 - (N209.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 254-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt. Nach Abarbeitung der Schleife wird mit Schritt 3 fortgefahren.**[<=]**

A_17016 - (N209.210) K_Performanztest
Schritt 1: Im *i*-ten Schleifendurchlauf gilt: *searchString* = I2OS(*i*, 1).**[<=]**

G2_N209.220 - (N209.220) K_Performanztest
Schritt 2: In *currentEF* MUSS mittels Use Case aus (N067.900) gesucht werden, wobei als Parameter *recordNumber* = 1 und als Datenteil der Kommandonachricht *searchString* aus Schritt 1 verwendet wird. Die Laufzeit $t_{search,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N209.230 - (N209.230) K_Performanztest

Schritt 3: In *currentEF* MUSS mittels Use Case aus (N067.900) gesucht werden, wobei als Parameter *recordNumber* = 1 und als Datenteil der Kommandonachricht *searchString* = '00' verwendet wird. Die Laufzeit$t_{search,255}$ dieses Kommandos MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17017 - (N209.240) K_Performanztest
Der Performanztest MUSS die Performanzpunkte $P_{terminateCard}$ wie folgt ermitteln:
$P_{SearchRec}$ = points(($t_{SearchRec,1}$, $t_{SearchRec,2}$, …, $t_{SearchRec,255}$), $T_{SearchRec}$)**[<=]**

Testnachbereitung:

G2_N209.300 - (N209.300) K_Performanztest
Aus / MF / DF.strukturiert / EF.strukturiert MÜSSEN mittels Use Case aus (N063.442) alle Rekords entfernt werden.**[<=]**

18.8.10 Symmetrische Sessionkeyaushandlung für Secure Messaging

In diesem Abschnitt wird lediglich die symmetrische Aushandlung von Sessionkeys für Secure Messaging betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln SK.AES128, SK.AES192 und SK.AES256 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

G2_N210.110 - (N210.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N210.120 - (N210.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.**[<=]**

G2_N210.130 - (N210.130) K_Performanztest
Schritt 3: Es MUSS eine leere Menge $M_{GetStatus}$ = {} erstellt werden.**[<=]**

Testdurchführung:

G2_N210.200 - (N210.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {SK.AES128, SK.AES192, SK.AES256} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 7 ausgeführt.**[<=]**

G2_N210.210 - (N210.210) K_Performanztest
Schritt 1: MSE Set Kommando gemäß (N102.400), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4SM gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N210.220 - (N210.220) K_Performanztest
Schritt 2: Der Performanztest MUSS im Rahmen der Testdurchführung eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 5 ausgeführt. Anschließend wird mit Schritt 6 fortgefahren.**[<=]**

G2_N210.230 - (N210.230) K_Performanztest
Schritt 3: Es wird eine Zufallszahl mittels Use Case aus (N098.625) vom Prüfling abgeholt. Die Laufzeit $t_{rand,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N210.240 - (N210.240) K_Performanztest

Schritt 4: Es wird eine erfolgreiche gegenseitige Authentisierung gemäß Use Case aus (N083.800) und (N084.410)a durchgeführt. Die Laufzeit $t_{Auth,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

A_17018 - (N210.250) K_Performanztest
Schritt 5: Die Ausführungszeiten MÜSSEN wie folgt zusammengefasst werden:
$t_{SK,i} = t_{rand,i} + t_{Auth,i}$. **[<=]**

(N210.260) Dieser Punkt ist absichtlich leer.

A_17019 - (N210.270) K_Performanztest
Schritt 6: MSE Restore Kommando gemäß (N099.900) mit *seNo* = 1, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

Testauswertung:

A_17020 - (N210.280) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{SK4SM,AES128}$ = points(($t_{SK128,1}$, $t_{SK128,2}$, …, $t_{SK128,100}$), $T_{SK4SM,AES128}$)

$P_{SK4SM,AES192}$ = points(($t_{SK192,1}$, $t_{SK192,2}$, …, $t_{SK192,100}$), $T_{SK4SM,AES192}$)

$P_{SK4SM,AES25}$ = points(($t_{SK56,1}$, $t_{SK256,2}$, …, $t_{SK256,100}$), $T_{SK4SM,AES256}$)

[<=]

Testnachbereitung:

Keine.

18.8.11 Schlüsselimport und asymmetrische Authentisierungsprotokolle

In diesem Kapitel wird der Import von Authentisierungsschlüsseln mittels CV–Zertifikaten behandelt, wobei die öffentlichen Schlüssel aller verwendeten CA bei Produktion des Prüflings bekannt sind. Zudem wird die Rollenüberprüfung (eine andere Komponente authentisiert sich gegenüber dem Prüfling) sowie die asymmetrische Aushandlung von Sessionkeys betrachtet.

Der Prüfpunkt beinhaltet einerseits:

1. Den Import eines Authentisierungsschlüssels.
2. Eine Rollenprüfung zur Erlangung eines Sicherheitszustandes.

Zudem wird hier ebenfalls geprüft:

3. Der asymmetrische Aufbau eines Trusted–Channels zur Karte.

18.8.11.1 ELC 256

Es werden zehn Test CA benutzt, denen von der Root–CA (siehe PuK.RCA_ELC256 in CosT_d98) folgende CV–Zertifikate zugeordnet werden:

Tabelle 356: CosT_ece: ELC-256 Sub-CA CV-Zertifikate für den Performanztest

ID	CV-Zertifikat
Test CA0	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 043dc8968bf65157711a3547714230e7d80667d1451897204a9fcd91e5e53b1aeb960a4257e 36656f06da638d50d0a4095fbfa11c99c8481da49fe5245a47638855f200844455858586f01 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f3740960632d2f613697ccbb05b3b21ae6ddb1fbb146deb5cc3f7787a28d485c2e7 042b5001db598eac960055c56e3489c568302ff3e638a8b5bece070995859629a9'
Test CA1	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 04068fb3f80118c9a1390b6f3d6d84a5dd5f5c1228e57268afd09297b71c5d8d451879b7097 be706fb9a9e6952f599ef0f7eaa13f111d9c1ec656fdc93163ed9575f200844455858586f11 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f37406fc05554871d4f0cc60386b500e21975471265ca08a47b932082b596e8be8d bc6e68a729051c47b38db23e16667ff4fae724e39529c7ebd8f05aab249e824c9b'
Test CA2	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 045fe0d65046d587cf0d494766ed0f4acd646f00820c7d98458df25f49aab891cd2b80e5d51 c18719354bf8c8f3a83a41ec8819e894a55a27b8acfe212393f4e685f200844455858586f21 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f37404f73ecc1733ce8acba3c539b45ba3ba79829a65d6d1057f3f3732fd32c1920 ca8efb4489072cc4943b4b948421b146053f39bd51c5275d534ec395333b9f6430'
Test CA3	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 04a2f12f29f8c45da68c2d9d484ef4f764b861934f19c8ad829edad541050b1ad9323d83083 7144714bac3ff9ffa9b9f2ec5288e93373744ba2923c7afb72e6fde5f200844455858586f31 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f374048a2a0e899ac08817249726f1f13bb983ee5728d3439869953b46b8b69c41d 02690ebe1be8a04a5897c82ca04eedc527a016d798f07a776d75fa99f2c7f88e7f'
Test CA4	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 04156028235673196c4d54e4b78205c9979d2bdb39362d05568ee5c11671ac9010284bb19e2 4444b6f174167cd50efd5e6ef9a5780788ac02312f6961dd180e9785f200844455858586f41 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f374081d4965d98f35605fcb6bb6303378787810af9182d2b9fb495b22161a29e55 3a60ea87f7182c96153dc7ef1dfae79e32ad5dae554eca1bc244ab73f57d76cf14'
Test CA5	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 0462082556bbab7c0387eb3ee7a2767a2277beaf9a02ccdc255a07baf97bd232a74c18c5848 88ac8f655c05d56ac7afe399c79bfb865e6fe71ee8a76f35257ff0d5f200844455858586f51 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f37408db6a883d4e575c1667edd7fc1b3e1b41e7a5d8f73c8ce81b29232f671f4b5 178ea62299d06eae35c09794233ad1979b2e263449dbf4c9d68d74331a9a3d94eb'

Test CA6	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 041932b906841a94133354ef5fb2878d29612e83e3f55b43aceb44d809b1eebecd6c0c515de ce9c483273b5f070c3ab2924a65e26900e73506ed46d96be09bffad5f200844455858586f61 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f3740948249b88e2d1db43237cdc5c814865aed1ac1f8685cfece3222c2bd169c23 c15fe199585d714e74fd468893299f739f80c916e57dff8ffb1747624809122fa3'
Test CA7	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 04827bc42213603730397a0d1a25981fbf3d4c4326ebf943784fba310eebd0af39153d457fb 56d7dacfe7d3acc0397fa29160c67f0fc11caf4d9074b79213355bc5f200844455858586f71 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f37402bca5198dceee074b1a1ed84dddb52c06ea11e190aaa938636844aaba27741 207a66ac35471fa81fca9d04dea2e410fcf1757328fe9d8deec645e5707bdb8cad'
Test CA8	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 0427ed1d352e39fcf029f07314414df5a2070587e316f42dee798cf117e9dbfabb56a77fb65 169d1014b950eeb12efdcdd6e63d44e351386457b53ec58375d10c65f200844455858586f81 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f3740953e492d4fcd165cf0aa658aa0c3ccecf4bf4877ec52acdb3eb1b2ba2810e8 f30928563f2ce1e3666dac086ef84c749c29d6dfb21b448c67b435fe85bdfd126d'
Test CA9	'7f2181d87f4e81915f2901704208444558585860001127f494d06082a8648ce3d0403028641 0437b7e999de3102cfab44162d204b1bcab1d495da3f3c1293db90198b2399f42920d2b716e 8e5fceeba0a760df6a27009b4f5b5a51a4dae74c4e0aa03d9ab16515f200844455858586f91 127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f240602030 00301085f374006c12b8bb2affd0e2dc64dc9a51df59ac7d0685b736c58a5af92c0e7c0147e 151f937657c6500bb3361c9b504ec39f3867990201ff78d238aa6e797aa74f59c7'

Test:

A_17021 - (N211.110) K_Performanztest
Der Performanztest MUSS für diesen Prüfpunkt mit folgenden Parametern arbeiten:
a. M_{CVC} = {TestCA0, TestCA1, …, TestCA9} aus CosT_ece
b. *PuK.RCA* = / MF / DF.Auth / PuK.RCA_ELC256
c. *PrK.Auth* = / MF / DF.Auth / PrK.Auth_ELC256**[<=]**

A_17022 - (N211.120) K_Performanztest
Der Performanztest MUSS die Performanzmessung für ELC-256 wie folgt durchführen:
(M_{Import}, $M_{RoleCheck}$, M_{SesKey}, $M_{GetSecSta}$) = PerformanceAuth(M_{CVC}, *PuK.RCA*, *PrK.Auth*) **[<=]**

Testauswertung:

A_17023 - (N211.130) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{Import,ELC256}$ = points(M_{Import} , $T_{Import,ELC256}$)

$P_{RoleCheck,ELC256}$ = points($M_{RoleCheck}$, $T_{RoleCheck,ELC256}$)

$P_{SesKey,ELC256}$ = points(M_{SesKey} , $T_{SesKey,ELC256}$)

$P_{GetSecStat,ELC256}$ = points($M_{GetSecStat}$, $T_{GetSecStat,ELC256}$)

[<=]

18.8.11.2 ELC 384

Es werden zehn Test CA benutzt, denen von der Root–CA (siehe PuK.RCA_ELC384 in CosT_969) folgende CV–Zertifikate zugeordnet werden:

Tabelle 357: CosT_811: ELC-384 Sub-CA CV-Zertifikate für den Performanztest

ID	CV-Zertifikat
Test CA0	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d04030386 610450130d536c93d3183e4e7271fd291f34be2dde0869749b19420833220f10d4d8ae61f65 67710c277fca109565dea64c184f6bdea3470c2082283fcf58ee436983d90f91e6325274728 2cbe4218c1befcfd8bf7178be5dc0d0dd9a558a999b63e5f200844455858586f02127f4c130 6082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f 37605aed2ebd167b7872df46b37644afdac7d836602365025bfdb58af88404c3986e97b0771 677aafc326bc46e2ae3ff6c72718ad232da064b7ff31aec2822d78dd8c4e56613e233a0e76f bf92ad2bf8d6704189a11910f5e94fb45436b578b0894c'
Test CA1	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d040303866 1048c24bff918480325f3556cf426460972c797e3ba9ea9c7c1ad0daccd29d3a09fcc9a4fc30 e7b0f23f99bb206c9a9a2722b9fa21bcd759368137edc309a31214d610cbe08ba9590215c646 cbb54b3548268884afc5e4f84c7fa86c8fcba091f595f200844455858586f12127f4c1306082 a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37603 48a55a41fe1624641874b5b10f31667885374ac9db23e67cc75f86895ffa279e444ecb7c0361 fabe13ba556056900524f309700775f9774de0fb78371c661902cc11f19bb059e8a32170afd0 6ce4e47c19ad6b1a59ef99b993159354d8d8f95'
Test CA2	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 047c07cbafca6624495a06ab1cc0bbc42210181cd0ba12eb687eb78dafd77a65ec8314aaafcad bada5c071ef705857c7644807136b106d07befc1be5279e7b4536bd7fc70016ce20b448479594 441e8179ee4eeda328cc5034756b6871eede6a6c5f200844455858586f22127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f376035f215 4ab00b3d4df8abbed23c261336a833fb977cc18f4dc570754bb89b91e62ffa1605609a0c687f6 4f88ea0bc83ad4c8c615cb76868b73f7e55bef0bb3e0db3043d386f1faf32e477a2f0803965bd a9ac9a5afecf8d29ae6edfdbd4bd2c3e'

Test CA3	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 046aec3041253a998dde1b7f060abeab85939b9c6ab6dd32d7f01b03736d22154c63753c375ae 7783189484be4c94772bc5e39cf3b02bf35218c0b1f63bfc234485bcdea8bfe55c39e32928be7 ca61638321b03b5dd1912960079138166943a0645f200844455858586f32127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37607aeedf 7e06c1f2a4042c3937cd0e8110b1a21bb3526f5363451afa4ae0ba67141c3448176c301d71988 2e67d8d04e9115f5b6ad5ab3ade0310445a5f314c20ab46a2ee4b03bf528f688e36689749dd55 0ed94c8cb4a9b12ff7e76eaef14cf846'
Test CA4	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 0454d62af2abd57a84f24fd73dfbf1d42074b097549de2f50b349b321866734cad23c2b3705bd fc54b51e4e5d214aa4fc753fab4c7d443af476ff1c8efd4ce6aa9db8e0b65f0f27bd8df797f36 dbb4cf6aade07ac21f5081fe22b497f1aefbce9f5f200844455858586f42127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f376015369d 871fb8a1fc6d1ff7c0f71289a07a04df5fe1df47da24e55cf475c47687239cf438668610a76c8 536a0085dfb0e0d51cafe24138e4834e38c3fe4e05b387e28e0cd0f13db19ed01ac612946c317 47d6a2616aa01ac8b69c7f3ff1f37dc5'
Test CA5	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 044844e9c051ddb7772cd5700eb9a192382f0b1fbf4a05ffca026510a8f3b70ef26437e842564 3ba3b4995e6b1da512c444cc44c14533f253b2507adfbb50544d5573823a4543156bbaf052e7d d54395284dded89b3829202212961c7b5e745ff95f200844455858586f52127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f376049b34f 8f26d761ee83be427566481c5270e1a8a0bad98e3a1ef3492794031195e550f9358f6323feff8 7916ea731936b5cabb3c29e57f2b637b91e22ca5cefd6e92348319ab46ae6825acf4c6b332db6 296f0d00cdce82b1eb65fc65b39f7aaf'
Test CA6	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 045c9983c0ad853bc555a63f94b3245b688f2ac8de637f5426f9064d00b2f987776da4a4f7c44 5184dce6e76d2e5ef56a40f8d9e3c895790271923b561dfc675819457fcd9edc23093a1962171 310ba8a9527ed54f7cac27475f37fad1a90e2d175f200844455858586f62127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37607c432e e2ec2640b13a6864ff201880c9392378983201c4abf8e1d55ed8a36c22ac7fd2b8bf8239f5f46 ac1a17d1136af2dbd3035979890828d201872af34ce07a6b862f7d5c0ed4ba8dafaa657443ecc 04e1b31772911c5459cf0d4c4eb51582'
Test CA7	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 045949a568bc6e923f5120d246272947cc9ae33a3095b70a95e469c0ffb1e4763b121ece59e33 f1275fb896f96e3e2c4752070bd62fae268693aa30bf995874302528074e32f3ba17f94433f0b 71576f19f05de2589b395a8bd22409e1ddc93b3a5f200844455858586f72127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f376039cd3a 5d6da66dfea52e514d06ef8186e4ddcdea98185764164a469f9d5f5a3315fb383b73c490d32d1 b0424a9ca5c8923ac63aa349cad6620f7c8ea4611ee2724ccf5e68a90bfe7fa008931ae8f7840 1ee00fc37c57ce1f5a7c8a69792063a1'

Test CA8	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 0479ebc9b3c7a373818959904d9154f63a4a08767955fdb23fa6225c39b70147ba5943facaa71 e1852b13ba870beb27dbd4dffaeac0198f5bdcb672b4da129778fb1eac47a75f79e4f4d55df70 6324d2b593ef00de4e2de19093f29f046906150a5f20084445585858 6f82127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37608299ce 56898da0b1b584130f11b03293f05278c9a3619b84f67fa12b9a972dd4001b013c78c2260a409 87f813e76f6e73122767f4030c2832a7260b9d2e8a39de6164b35804735de29b65751006b0847 90ee68fdc08c792664ff4a89e3859cad'
Test CA9	'7f218201187f4e81b15f2901704208444558585860021 27f496d06082a8648ce3d0403038661 041fce13dbe6187eca90441f24f69059e97741ee761c934f0954fb16abf3070d80cf974910ce8 51d6fcbfa50a10b78de4c5586c538dccf5633152c32e158f22d6cf60b0562e509f1b65caf5fc4 3641cf952917c1d6eed4736d43029a5372bb85445f200844455858586f92127f4c1306082a821 4004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37602c787c e37d15b432af5cf70656900958c705d0ce025bd9a56075faed3ae7fc345f0c03b865787c487b1 ff6730b50a7ee0b26245a89b31105641fe9c183297ac776147c5e572e271dbe3fd17d199a2bc1 f94f7920e1c670daf2035108089e26e8'

Test:

A_17024 - (N211.210) K_Performanztest
Der Performanztest MUSS für diesen Prüfpunkt mit folgenden Parametern arbeiten:
a. M_{CVC} = {TestCA0, TestCA1, …, TestCA9} aus CosT_811
b. *PuK.RCA* = / MF / DF.Auth / PuK.RCA_ELC384
c. *PrK.Auth* = / MF / DF.Auth / PrK.Auth_ELC384**[<=]**

A_17025 - (N211.220) K_Performanztest
Der Performanztest MUSS die Performanzmessung für ELC-384 wie folgt durchführen:
(M_{Import}, $M_{RoleCheck}$, M_{SesKey}, $M_{GetSecSta}$) = PerformanceAuth(M_{CVC}, *PuK.RCA*, *PrK.Auth*) **[<=]**

Testauswertung:

A_17026 - (N211.230) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{Import,ELC384}$ = points(M_{Import} , $T_{Import,ELC384}$)

$P_{RoleCheck,ELC384}$ = points($M_{RoleCheck}$, $T_{RoleCheck,ELC384}$)

$P_{SesKey,ELC384}$ = points(M_{SesKey} , $T_{SesKey,ELC384}$)

[<=]

Hinweis CosH_f07: Das Ergebnis $M_{GetSecSta}$ *wird hier nicht weiter verwendet, weil dieser Prüfpunkt bereits in 18.8.11.1 betrachtet wird.*

18.8.11.3 ELC 512

Es werden zehn Test CA benutzt, denen von der Root–CA (siehe PuK.RCA_ELC512 in CosT_1db) folgende CV–Zertifikate zugeordnet werden:

Tabelle 358: CosT_649: ELC-384 Sub-CA CV-Zertifikate für den Performanztest

ID	CV-Zertifikat
Test CA0	'7f2182015b7f4e81d35f2901704208444558585860003127f49818e06082a8648ce3d040304 8681810495fc9b7543c2ff1ed4394027ad2acb94358fb6e82654e7157fd7c53c82bee4418c1 f466c5c9ee3e7a10a5bb82947d97ae11690c784c216960a0ae68e300b449950ba3b9be3eb68 8e8c0b6d0b4fb03f5fbceacb21a00c31da9fb7f2c319e679d0d006739e18e80fca70c8702ae a4cd9d986fd84896e07ee851815170d7a5c95825f200844455858586f03127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781800 7b9a510c34a95f6fe3a5334db0145faa7678d524067a31b9817125724c5e278424dd6fa3a7b e20f4f762eb0938972b76195618445cb0c12a29b2f4693865b9d84d6d813641f4911f34b5d3 f7b9abccde60070a2fc85b2ac0c5d7ab44051b0f959399cf49a8beef2be13f934d3f976dca3 877eb69e8499e4a28b5a3d0279a2da'
Test CA1	'7f2182015b7f4e81d35f2901704208444558585860003127f49818e06082a8648ce3d040304 868181045f9a82a8f06b6ba10e32f9a21c002c4b96fde78bf47855d26aa4f8d52194756b7ec 537559d6a1c6e6a9d12c0634733279e862ce8c50cb68024898dc675d4b52359fdefffbf2d39 7d8f7e4f0365e0f13ddb633c2d6311205a268e502921249e429b73e1defdf5d834531d352b2 5a47162792d4c6096682ddbe36b11e54875689e5f200844455858586f13127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781804 e92e250b06ce31d1ace367543fe3cba02b3d9b938397112a84f8c83a70f701acedefb94a047 892ce60d464e3cb22ddaf0ca1ce2c964248a5b5619b7ea71f47f63e2343584b52290dbb46a3 c599ed7f5b2344acc3cf1bb9843999247061c0a76350585e0fb8ab14424b0fc5539c00d9999 1ce856a83bd1d57da069ea2b8c5a87'
Test CA2	'7f2182015b7f4e81d35f2901704208444558585860003127f49818e06082a8648ce3d040304 868181046038d73f8444a04b965a541af9247fef5d639be21793bb4bca616c773ac5dced51c ee5b36220fdbb2a4f320fbd5143fda7700cf17a2a0b49d03c9157ec9960309331754283ef76 f0db10cb599397cf8da23535857bb387ba097696f07020c75df500c6521b749fc53f38a57a5 62f18c2b5145ded45b6dd9ce7274a60bfe9d12e5f200844455858586f23127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781809 835527c0fa6f50056296af1dbe72c3f0836820f4be045d8f147048ba622ae785f58ead4118b 4d43f9c69899ca4c87d84d45c487ca309e0b50aa5d8513553d17941bd3cc90ee467347bdf0e 79e9db5dbb8ce16f5ad68dfb2227cdfe42b0945dde03b87394b70c32c9c51763340e2f09fa8 5426552d3b0a0b8104c52cef8abf15'

Test CA3	'7f2182015b7f4e81d35f2901704208444558585860031 27f49818e06082a8648ce3d040304 8681810499ca069ab8c0228c85b4c0d0bbbf37d833d658b9ce81b0adc5b762224fd0011801f 321435df34677ccd21f6058832fc0b372df96b24ac4cb970681d381f5bc5c22d8a854adbe0b 9ddd15bb049589d1b1360816ced81c8cdf43106133b2612710a67024ae0e563f91e849aaf63 acc572dc48b9dcd6f87ce5cc203500dd0c638e55f200844455858586f33127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781805 f21aae86591351d5ddca7634eaa8a9f12e69f50a87d1d8a4835c24720098e66f1371262b4ca b80733a7e4ea28c9f180852eb19b4e2b0f3db0cb873ddfa92bf75acb2d072b9c810d415a7b6 73fd229ad16adeb23ecb2fb0fc7dca8a565fea391ff4d5b487fa3e5c37fb8d5775599586b04 4bae6589246b88c3215d05a1c5b309'
Test CA4	'7f2182015b7f4e81d35f2901704208444558585860031 27f49818e06082a8648ce3d040304 8681810431a80097158da05ecfea043eba55deead39c6e0a078c76e6e668a224ac4b7717213 b56521df9b4d77006aff6a5ffe1499070f33bb261a31f675631d6075df9be63d9083b253189 1a995accadbeca7cf20dbc3db15d7b1bf545b5556d6fc7a889c0ff25c04bd2e186955d65625 7c936cc1e742b49de7ea1e8879b915e512cad545f200844455858586f43127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781809 99c42b069726e175b059480e2458ae101de5b3f842187ab8e11bc9b26651b729c8eba497fbc 56a6dd1259794ef530db11af9022aff968d237f050e4684935a89f49a69108492e71df26176 3d9ba1a46ae9e41bbe205906fb602f9f6b5d3407741f6f482daaa789c2e5131e95c09544664 7ea8dc75e40386248012c18a915e39'
Test CA5	'7f2182015b7f4e81d35f2901704208444558585860031 27f49818e06082a8648ce3d040304 8681810451e931e921d6d7a771bc53e6c7770ed184817b5424a9918761046f34348d45a0677 13c04895e68c8cefc140e0a63b49703df483ba9a39f3fb06ee5e8eb4aba651cf33120c97df1 4dbfa34d13639ec70b11bbc6671e7e6f756705dd4365dc22e67bcad2e09b013ff9dacaadbec aef90069ca481cf9ab6e61f0f681e4ef91247845f200844455858586f53127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781809 1fea03f291142ff6b9942ef1b9c5fa191d53769ad1f156438815dd92438e6354b00ab9447f6 ef79cedfbe51de1de8e4873e3982afb65261be556154059299a0058ebdb695d549213bb3d2a aaecc738f92cfd9c1e561e376b098ac4ec4dfc2d24075c62688681f366dc4e7ca5bd36c8cba 61e8191aace6090ac83651b87bd309'
Test CA6	'7f2182015b7f4e81d35f2901704208444558585860031 27f49818e06082a8648ce3d040304 868181042edd3f3057e2cb122862bd3999e9368136d34330739a2a667720d1ea48b45fd7050 b9645b099382c85759ae07074f76ea827c716d300700fd69b57dab282a4fa5bb4b1f62b19e9 9cccfe5a3f0d24cd6937c31d28903dfcf5bc827bf8bd8c7502607f60b510d2c893663247b0b 40d3f1d68bffc4e7f1aa27daeb745668ab3e0725f200844455858586f63127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781808 c8203e3edca689def8a349e4cc0e3fad9aca951f63f4d0aa215a5333bd93b17795ccdbcf271 f7fa243d60472fcfe96be7b8ae7673a4b375436856d02d6df6c82afa534f533ba2831672747 8dabe5f4ecc215c442d4be6ac7a3e38890bbc8d758d8d75ada0d2c820c379b6b8744ecb9491 d5010258a5cab4d929ba9e2f5cb9ee'

Test CA7	'7f2182015b7f4e81d35f2901704208444558585860031 27f49818e06082a8648ce3d040304 8681810414040402806884770 3b9ff2f2bd45927333e36da543533ff51d440cdbed8ddd5265 23524b89cd03320b8eb04593096c78cd0ab6ff4c3711fee04284d2e6bb003672d53f7111e05 a14883e93482570667e62bd10c39c9999186d8486eba560abd2308275113861b6e74cf5968c c20bad34af21f74a4d54c985bcbcb3af9c88de05f200844455858586f73127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781809 622b1a6b7be6087964877bf8d285e98a6d1f4defcb34eb48a34c7b0685fc8da3a33d8bff8b2 b543c9567410da87ce8233f3d56d06b2d021beb30865b1d1193d87f835d12de3b5c89fefa45 ca40378bc96d046f926c3d305ef888570f92d9c7ce53ffde6439a91b4e4be2a42208cfaadcb 4661be54ab7b6300f2e1da47fde34a'
Test CA8	'7f2182015b7f4e81d35f2901704208444558585860031 27f49818e06082a8648ce3d040304 8681810424 16f09a534980a619c4a121687e82e5af1731f8de2c807c5eb00b4f458793b0ee2 fa2b1e134d8aad1d1437a5fbeeb4c425e7b56d90d369f214de22ce41b1d5c8ec7d75a8b2546 4fe7fcca3ce4e713cb5d3addfc418ef2741a37a256e0b30f231688cc6c4705827e92f995e1c d9ade79129d32f570af100b7e95db030b88e7105f200844455858586f83127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3781802 717e550c4f907ad7a652904f960b3ea42eb03a689848556460e79fa60de3c8d1d8f2285b3a0 6a5cd17c12005ff4658194e819ac9cc715a0f56f4f7ac64e02f4073416c26a8a574a21c0826 2eda700036320d3492a80632c14a090a39533a8fcee7dd67c6f8a98adf04e5c1df03f14eb49 ace6a6621aec24937fd56fad1be374'
Test CA9	'7f2182015b7f4e81d35f2901704208444558585860031 27f49818e06082a8648ce3d040304 868181049bdab8d4c7e9fd163c9cfcbf653909d2cef4032d1b0110990af18d14a538b30b3b7 ff3abde1180ba8b07094eec56deda3e044611acab2ebc52f2da9bf8a08c894b1edfcf679b4c 259b871dfd582588c81df4c9f0fe478bebf5d322f19ec843ae3aa704964775228ece5ad3092 7029f6f53cfcfc3663f48a4f4ddf7ad30f6c5125f200844455858586f93127f4c1306082a82 14004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f378180a 7cd37e35558fb279ef6acaf57aea6efc5d5d120e192dab917c7824d4fe4ea8b0a0b723add67 1c671704d667da2f9afc91522c06a175b3577370161e1b69bb399253b362661dab2cfde3c99 091084a8b60b5c3132848e2c2962c1560ddc073143e2f8fec374d29f7f61c9cb621e15bfa00 df40be5bb202f8f85554158ac7a965'

Test:

A_17027 - (N211.310) K_Performanztest

Der Performanztest MUSS für diesen Prüfpunkt mit folgenden Parametern arbeiten:

a. M_{CVC} = {TestCA0, TestCA1, …, TestCA9} aus CosT_649

b. *PuK.RCA* = / MF / DF.Auth / PuK.RCA_ELC512

c. *PrK.Auth* = / MF / DF.Auth / PrK.Auth_ELC512**[<=]**

A_17028 - (N211.320) K_Performanztest

Der Performanztest MUSS die Performanzmessung für ELC-512 wie folgt durchgeführen:

(M_{Import}, $M_{RoleCheck}$, M_{SesKey}, $M_{GetSecSta}$) = PerformanceAuth(M_{CVC}, *PuK.RCA*, *PrK.Auth*) **[<=]**

Testauswertung:

A_17029 - (N211.330) K_Performanztest

Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{Import,ELC512}$ = points(M_{Import} , $T_{Import,ELC512}$)

$P_{RoleCheck,ELC512}$ = points($M_{RoleCheck}$, $T_{RoleCheck,ELC512}$)

$P_{SesKey,ELC512}$ = points(M_{SesKey} , $T_{SesKey,ELC512}$)

[<=]

Hinweis CosH_58f: Das Ergebnis $M_{GetSecSta}$ *wird hier nicht weiter verwendet, weil dieser Prüfpunkt bereits in 18.8.11.1 betrachtet wird.*

18.8.11.4 RSA 2048

Dieses Kapitel ist absichtlich leer.

18.8.11.5 Testablauf Schlüsselimport und asymmetrische Authentisierung

Dieser Abschnitt beschreibt auf generischer Ebene den Ablauf der Performanzmessung für Schlüsselimport (PSO Verify Certificate) sowie für eine asymmetrische Authentisierung mit und ohne Sessionkeyaushandlung. Der Ablauf wird für ELC-Schlüssel unterschiedlicher Länge durchlaufen.

Tabelle 359: CosT_b22: Definition der Funktion PerformanceAuth(...)

Input:	M_{CVC}	Menge von CV-Zertifikaten die einer CA zugeordnet sind
	PuK.RCA	Sicherheitsanker der PKI, mit welchem sich die Zertifikate in M_{CVC} prüfen lassen.
	PrK.Auth	privater Schlüssel des Prüflings, der im Rahmen der Sessionkeyaushandlung benötigt wird
Output:	M_{Import}	Tupel mit Ausführungszeiten zum Schlüsselimport
	$M_{RoleCheck}$	Tupel mit Ausführungszeiten zur Rollenüberprüfung
	M_{SesKey}	Tupel mit Ausführungszeiten zur Sessionkeyaushandlung
	$M_{GetSecSta}$	Tupel mit Ausführungszeiten zur Abfrage Sicherheitszustand
Errors:	-	keine
Notation:		(M_{Import}, $M_{RoleCheck}$, M_{SesKey}, $M_{GetSecSta}$) = PerformanceAuth(M_{CVC}, *PuK.RCA*, *PrK.Auth*)

Testvorbereitung:

Jede Test_CA aus M_{CVC} erzeugt zehn CV–End-Entity Zertifikate. Insgesamt ergeben sich so einhundert CV–Zertifikate mit öffentlichen Authentisierungsschlüsseln.

G2_N211.510 - (N211.510) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N211.520 - (N211.520) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.**[<=]**

Testdurchführung:

G2_N211.600 - (N211.600) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100–mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 0 bis 12 ausgeführt.**[<=]**

A_17030 - (N211.602) K_Performanztest
Schritt 0: MANAGE SECURITY ENVIRONMENT Restore Kommando gemäß (N099.900) mit *seNo* = 1. Dadurch werden alle Elemente aus der Liste *dfSpecificSecurityList* (siehe (N029.900)) entfernt. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N211.610 - (N211.610) K_Performanztest
Schritt 1: Aus den einhundert CV–Zertifikaten mit Authentisierungsschlüssel wird ein bislang noch nicht verwendetes gezogen (ziehen ohne zurücklegen). Durch die Ziehung wird folgende CV–Zertifikatskette gebildet:
PuK.RCA → CVC_Test_CAx → CVC_ICCy.
Die erste Ziehung ist beliebig. Bei allen weiteren Ziehungen MUSS die Nebenbedingung beachtet werden, dass CVC_Test_CAx verschieden ist vom unmittelbar vorher verwendeten Zertifikat CVC_Test_CAx. ~~Dann MUSS mit Schritt 4 fortgefahren werden.~~ Dann MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N103.300), wobei als keyRef der Wert CAR aus CVC_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der i-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run1.\,i}$ bezeichnet. Wenn dieses Kommando mit NoError beendet wird, fahre mit Schritt 5 fort, sonst mit Schritt 2.**[<=]**

G2_N211.620 - (N211.620) K_Performanztest
Schritt 2: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N103.300), wobei als *keyRef* der Wert *keyIdentifier* aus *PuK.RCA* verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run2,i}$ bezeichnet. **[<=]**

G2_N211.630 - (N211.630) K_Performanztest
Schritt 3: Wenn das im vorherigen Schritt selektierte Schlüsselobjekt ein

a. ELC Schlüssel ist, dann wird ein PSO Verify Certificate Kommando gemäß (N095.500) verwendet, wobei als Parameter *certificate* CVC_Test_CAx verwendet wird.
b. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run3,i}$ bezeichnet.

[<=]

G2_N211.640 - (N211.640) K_Performanztest
Schritt 4: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N103.300), wobei als *keyRef* der Wert CAR aus CVC_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run4,i}$ bezeichnet. ~~Wenn dieses Kommando nicht mit NoError beendet wird, dann fahre mit Schritt 2 fort, sonst mit Schritt 5.~~**[<=]**

G2_N211.650 - (N211.650) K_Performanztest
Schritt 5: PSO Verify Certificate Kommando gemäß (N095.500), wobei als Parameter *certificate* CVC_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run5,i}$ bezeichnet. **[<=]**

G2_N211.660 - (N211.660) K_Performanztest
Schritt 6: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N101.900), wobei als *keyRef* der Wert CHR aus CVC_ICCy und als *algId* elcRoleCheck verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run6,i}$ bezeichnet. **[<=]**

G2_N211.670 - (N211.670) K_Performanztest
Schritt 7: Wenn im vorherigen Schritt

c. elcRoleCheck verwendet wurde, dann sende ein GET CHALLENGE Kommando gemäß (N098.625).

d. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run7,i}$ bezeichnet.

[<=]

G2_N211.680 - (N211.680) K_Performanztest
Schritt 8: EXTERNAL AUTHENTICATE Kommando gemäß (N083.500) und (N084.400)a~~, b)~~. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run8,i}$ bezeichnet. **[<=]**

G2_N211.690 - (N211.690) K_Performanztest
Schritt 9: Abfrage eines Sicherheitsstatus:

1. Wenn M_{CVC} ELC Schlüssel enthält, dann werden folgende Schritte ausgeführt:
   a. GET SECURITY STATUS KEY gemäß (N085.444) mit *oid* = oid_cvc_fl_ti und *cmdData* = '0FFF…FF'. Die Laufzeit dieses Kommandos in der*i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run9.1,i}$ bezeichnet.
   b. GET SECURITY STATUS KEY gemäß (N085.444) mit *oid* = oid_cvc_fl_ti und *cmdData* = *flagList* aus CVC_ICCy. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run9.2,i}$ bezeichnet.
2. Es gilt: $t_{GetSecStat,i} = t_{Run9.1,i} + t_{Run9.2,i}$.

[<=]

A_17031 - (N211.700) K_Performanztest
Schritt 10: MANAGE SECURITY ENVIRONMENT Restore Kommando gemäß (N099.900) mit *seNo* = 1. Dadurch werden alle Elemente aus der Liste *dfSpecificSecurityList* entfernt. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N211.710 - (N211.710) K_Performanztest
Schritt 11: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N100.900), wobei als *keyRef* der Wert *keyIdentifier* von *PrK.Auth* und als *algId* elcSessionkey4SM verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run11,i}$ bezeichnet. **[<=]**

G2_N211.720.a - (N211.720)a K_Performanztest
Schritt 12: Aushandlung von Sessionkeys:

a. Wenn M_{CVC} ELC Schlüssel enthält, dann werden folgende Schritte ausgeführt:

i. GENERAL AUTHENTICATE gemäß (N085.012), wobei als *keyRef* der Wert CHR aus CVC_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run12.1,i}$ bezeichnet.

ii. GENERAL AUTHENTICATE gemäß (N085.016), wobei als *ephemeralPK_oponent* die Antwort des vorherigen Kommandos verwendet wird. Die Laufzeit dieses Kommandos in der *i*–ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und wird mit $t_{Run12.2,i}$ bezeichnet.

iii. Es gilt: $t_{Run12,i} = t_{Run12.1,i} + t_{Run12.2,i}$.

[<=]

Testauswertung:

A_17032 - (N211.730) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:
Bei der Bearbeitung einer Schleifeniteration sind folgende Fälle denkbar:

1. Der öffentliche Schlüssel der CA ist bereits in der Karte gespeichert, weil er bei der Kartenproduktion in *persistentPublicKeyList* oder durch einen früheren Zertifikatsimport in *persistentCache* gespeichert wurde. In diesem Fall besteht der *i*-te Schleifendurchlauf aus der Schrittfolge 1, 5, 6, 7, …. Die Laufzeiten der Schritte 1 und 5 werden summiert zu $t_{Import,i}$.
2. Der öffentliche Schlüssel der CA ist nicht in der Karte gespeichert ~~und dies wird bereits während der Schlüsselselektion bemerkt~~. Dann besteht der *i*-te Schleifendurchlauf aus der Schrittfolge 1, 2, 3, 4, 5, 6, 7, …. Die Laufzeiten der Schritte 1, 2, 3, 4 und 5 werden summiert zu $t_{Import,i}$.
3. Die Laufzeiten der Schritte 6, 7 und 8 werden summiert zu $t_{RoleCheck,i}$.
4. Die Laufzeiten der Schritte 11 und 12 werden aufsummiert zu $t_{SK,\ i}$.
5. Die gemessenen Zeiten werden zu folgenden Tupeln zusammengefasst:

$$M_{Import} = (t_{Import,1}, t_{Import,2}, \ldots, t_{Import,100})$$
$$M_{RoleCheck} = (t_{RoleCheck,1}, t_{RoleCheck,2}, \ldots, t_{RoleCheck,100})$$
$$M_{SesKey} = (t_{SesKey,1}, t_{SesKey,2}, \ldots, t_{SesKey,100})$$
$$M_{GetSecStat} = (t_{GetSecStat,1}, t_{GetSecStat,2}, \ldots, t_{GetSecStat,100})$$

[<=]

18.8.12 INTERNAL AUTHENTICATE zur Rollenauthentisierung

In diesem Abschnitt wird die Rollenauthentisierung mit privaten ELC Schlüsseln betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.Auth_ELC256, PrK.Auth_ELC384 und PrK.Auth_ELC512 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

G2_N212.110 - (N212.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N212.120 - (N212.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.**[<=]**

Testdurchführung:

G2_N212.200 - (N212.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.Auth_ELC256, PrK.Auth_ELC384, PrK.Auth_ELC512} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.**[<=]**

A_17033 - (N212.210) K_Performanztest
Schritt 1: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N100.900), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = eIcRoleAuthentication gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N212.220 - (N212.220) K_Performanztest
Schritt 2: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 4 ausgeführt.**[<=]**

A_17034 - (N212.230) K_Performanztest
Schritt 3: Es wird eine Zufallszahl *token* = RAND(16) erzeugt.**[<=]**

G2_N212.240 - (N212.240) K_Performanztest
Schritt 4: INTERNAL AUTHENTICATE gemäß (N086.400) und (N086.900)a~~, c~~. Die Laufzeit $t_{Auth,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17035 - (N212.250) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$$P_{RoleAuth,ELC256} = \text{points}(t_{Auth,1}, t_{Auth,2}, \ldots, t_{Auth,100}), T_{roleAuth,ELC256})$$

$$P_{RoleAuth,ELC384} = \text{points}(t_{Auth,1}, t_{Auth,2}, \ldots, t_{Auth,100}), T_{roleAuth,ELC384})$$

$$P_{RoleAuth,ELC512} = \text{points}(t_{Auth,1}, t_{Auth,2}, \ldots, t_{Auth,100}), T_{roleAuth,ELC512})$$

[<=]

Testnachbereitung:

Keine.

18.8.13 PSO Compute Digital Signature mittels signPSS

In diesem Abschnitt wird die Signaturberechnung mit privaten RSA Schlüsseln und dem Signaturverfahren signPSS betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509_RSA2048 und PrK.X509_RSA3072 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

G2_N213.110 - (N213.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N213.120 - (N213.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.**[<=]**

(N213.130) Dieser Anforderung ist absichtlich leer.

(N213.140) Dieser Anforderung ist absichtlich leer.

G2_N213.130 - (N213.130)
~~Schritt 3: GENERATE ASYMMETRIC KEY PAIR Kommando gemäß 14.9.3.10, wobei als *keyRef* = '8B' aus dem Attribut*keyIdentifier* von PrK.X509_RSA2048 gebildet wird und *operationMode* = 'C0' gesetzt wird.~~**[<=]**

G2_N213.140 - (N213.140)
~~Schritt 4: GENERATE ASYMMETRIC KEY PAIR Kommando gemäß 14.9.3.10, wobei als *keyRef* = '8C' aus dem Attribut*keyIdentifier* von PrK.X509_RSA3072 gebildet wird und *operationMode* = 'C0' gesetzt wird.~~**[<=]**

Testdurchführung:

G2_N213.200 - (N213.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509_RSA2048, PrK.X509_RSA3072} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.**[<=]**

A_17036 - (N213.220) K_Performanztest
Schritt 1: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N102.900), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = signPSS gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N213.230 - (N213.230) K_Performanztest
Schritt 2: Die Testdurchführung MUSS eine innere Schleife 64-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 4 ausgeführt.**[<=]**

G2_N213.240 - (N213.240) K_Performanztest
Schritt 3: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *dataToBeSigned* = RAND(*i*) erzeugt werden.**[<=]**

G2_N213.250 - (N213.250) K_Performanztest
Schritt 4: PSO Compute Digital Signature gemäß (N87.500) und (N088.600)a. Die Laufzeit $t_{run,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17037 - (N213.260) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{signPSS,2048}$ = points($t_{run,1}$, $t_{run,2}$, …, $t_{run,64}$), $T_{signPss,RSA2048}$)

$P_{signPSS,3072}$ = points($t_{run,1}$, $t_{run,2}$, …, $t_{run,64}$), $T_{signPss,RSA3072}$)

[<=]

Testnachbereitung:

Keine.

18.8.14 Signaturerzeugung und -verifikation mittels signECDSA

In diesem Abschnitt wird die Signaturberechnung und -verifikation mit privaten ELC-Schlüsseln betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509_ELC256, PrK.X509_ELC384 und PrK.X509_ELC512 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

G2_N214.110 - (N214.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N214.120 - (N214.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.**[<=]**

Testdurchführung:

G2_N214.200 - (N214.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509_ELC256, PrK.X509_ELC384, PrK.X509_ELC512} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 6 ausgeführt.**[<=]**

A_17038 - (N214.210) K_Performanztest
Schritt 1: GENERATE ASYMMETRIC KEY PAIR Kommando gemäß (N097.266), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *operationMode* = 'C0' gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

A_17039 - (N214.220) K_Performanztest
Schritt 2: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N102.900), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = signECDSA gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N214.230 - (N214.230) K_Performanztest
Schritt 3: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 4 bis 6 ausgeführt.**[<=]**

G2_N214.240 - (N214.240) K_Performanztest
Schritt 4: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *dataToBeSigned* = RAND(*domainParameter*. t / 8) erzeugt werden.**[<=]**

G2_N214.250 - (N214.250) K_Performanztest
Schritt 5: PSO Compute Digital Signature gemäß (N087.500) und (N088.600)c. Die Laufzeit $t_{sign,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N214.260 - (N214.260) K_Performanztest
Schritt 6: PSO Verify Digital Signature gemäß (N096.388) zur erfolgreichen Verifikation der im vorherigen Schritt erzeugten Signatur. Die Laufzeit $t_{verify,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17040 - (N214.270) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{signECDSA,256}$ = points(($t_{sign,1}$, $t_{sign,2}$, …, $t_{sign,100}$), $T_{signECDSA,ELC256}$)

$P_{signECDSA,384}$ = points(($t_{sign,1}$, $t_{sign,2}$, …, $t_{sign,100}$), $T_{signECDSA,ELC384}$)

$P_{signECDSA,512}$ = points(($t_{sign,1}$, $t_{sign,2}$, …, $t_{sign,100}$), $T_{signECDSA,ELC512}$)

$P_{verifyECDSA,256}$ = points(($t_{verify,1}$, $t_{verify,2}$, …, $t_{verify,100}$), $T_{verifyECDSA,ELC256}$)

$P_{verifyECDSA,384}$ = points(($t_{verify,1}$, $t_{verify,2}$, …, $t_{verify,100}$), $T_{verifyECDSA,ELC384}$)

$P_{verifyECDSA,512}$ = points(($t_{verify,1}$, $t_{verify,2}$, …, $t_{verify,100}$), $T_{verifyECDSA,ELC512}$)

[<=]

Testnachbereitung:

Keine.

18.8.15 PSO Encipher und PSO Decipher mittels rsaDecipherOaep

In diesem Abschnitt wird die Ver- und Entschlüsselung mit privaten RSA Schlüsseln und dem Entschlüsselungsverfahren rsaDecipherOaep betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509_RSA2048 und PrK.X509_RSA3072 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

G2_N215.110 - (N215.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N215.120 - (N215.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.**[<=]**

Testdurchführung:

G2_N215.200 - (N215.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509_RSA2048, PrK.X509_RSA3072} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 6 ausgeführt.**[<=]**

A_17041 - (N215.210) K_Performanztest

Schritt 1: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N103.800), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = rsaDecipherOaep gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N215.220 - (N215.220) K_Performanztest
Schritt 2: Die Testdurchführung MUSS eine innere Schleife 190-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 5 ausgeführt.**[<=]**

(N215.230) Schritt 3: Dieser Schritt enthält absichtlich keine Aktion.

G2_N215.230 - (N215.230) K_Performanztest
~~Schritt 3: GENERATE ASYMMETRIC KEY PAIR Kommando gemäß 14.9.3.10, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *operationMode* = 'C0' gesetzt wird. Die Laufzeit$t_{GAKP,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.~~**[<=]**

G2_N215.240 - (N215.240) K_Performanztest
Schritt 4: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *M* = RAND(*i*) erzeugt werden.**[<=]**

G2_N215.250 - (N215.250) K_Performanztest
Schritt 5: Wenn *keyRef* gleich dem *keyIdentifier* von

a. PrK.X509_RSA2048 ist, dann PSO Encipher gemäß (N090.790) und (N091.700)c, wobei *PuK* der öffentlich Schlüssel ist, ~~der in Schritt 3 erzeugt wurde~~ der zum in Schritt 1 selektierten privaten Schlüsselobject gehört, und *algID* = rsaEncipherOaep. Die Laufzeit $t_{enc,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.

b. PrK.X509_RSA3072 ist, dann wird *M* außerhalb des Prüflings äquivalent zu PSO Encipher gemäß (N090.790) und (N091.700)c verschlüsselt, wobei *PuK* der öffentlich Schlüssel ist, ~~der in Schritt 3 erzeugt wurde~~ der zum in Schritt 1 selektierten privaten Schlüsselobject gehört, und *algID* = rsaEncipherOaep.

[<=]

G2_N215.260 - (N215.260) K_Performanztest
Schritt 6: PSO Decipher gemäß (N089.200) und (N090.300)b. Die Laufzeit $t_{dec,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17042 - (N215.270) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{enc,2048}$ = points(($t_{enc,1}$, $t_{enc,2}$, …, $t_{enc,190}$), $T_{enc,RSA2048}$)

$P_{dec,2048}$ = points(($t_{dec,1}$, $t_{dec,2}$, …, $t_{dec,190}$), $T_{dec,RSA2048}$)

~~$P_{GAKP,2048}$ = points(($t_{GAKP,1}$, $t_{GAKP,2}$, …, $t_{GAKP,190}$), $T_{GAKP,RSA2048}$)~~

$P_{dec,3072}$ = points(($t_{dec,1}$, $t_{dec,2}$, …, $t_{dec,190}$), $T_{dec,RSA3072}$)

~~$P_{GAKP,3072}$ = points(($t_{GAKP,1}$, $t_{GAKP,2}$, …, $t_{GAKP,190}$), $T_{GAKP,RSA3072}$)~~

[<=]

Testnachbereitung:

Keine.

18.8.16 PSO Encipher und PSO Decipher mittels elcSharedSecretCalculation

In diesem Abschnitt wird die Ver- und Entschlüsselung mit privaten ELC Schlüsseln und dem Entschlüsselungsverfahren elcSharedSecretCalculation betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln PrK.X509_ELC256, PrK.X509_ELC384 und PrK.X509_ELC512 in der Anwendung / MF / DF.IAS.

Testvorbereitung:

G2_N216.110 - (N216.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N216.120 - (N216.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.**[<=]**

Testdurchführung:

G2_N216.200 - (N216.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {PrK.X509_ELC256, PrK.X509_ELC384, PrK.X509_ELC512} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 6 ausgeführt.**[<=]**

A_17043 - (N216.210) K_Performanztest
Schritt 1: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N103.800), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = elcSharedSecretCalculation gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N216.220 - (N216.220) K_Performanztest
Schritt 2: Die Testdurchführung MUSS eine innere Schleife 256-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 5 ausgeführt.**[<=]**

G2_N216.230 - (N216.230) K_Performanztest

Schritt 3: GENERATE ASYMMETRIC KEY PAIR Kommando gemäß (N097.266), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *operationMode* = 'C0' gesetzt wird. Die Laufzeit$t_{GAKP,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N216.240 - (N216.240) K_Performanztest
Schritt 4: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *M* = RAND(*i*) erzeugt.**[<=]**

G2_N216.250 - (N216.250) K_Performanztest
Schritt 5: PSO Encipher gemäß (N091.400), wobei PO_B der öffentlich Schlüssel ist, der in Schritt 3 erzeugt wurde. Die Laufzeit $t_{enc,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N216.260 - (N216.260) K_Performanztest
Schritt 6: PSO Decipher gemäß (N089.800). Die Laufzeit $t_{dec,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17044 - (N216.270) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{enc,256}$ = points(($t_{enc,1}$, $t_{enc,2}$, …, $t_{enc,256}$), $T_{enc,ELC256}$)

$P_{dec,256}$ = points(($t_{dec,1}$, $t_{dec,2}$, …, $t_{dec,256}$), $T_{dec,ELC256}$)

$P_{GAKP,256}$ = points(($t_{GAKP,1}$, $t_{GAKP,2}$, …, $t_{GAKP,256}$), $T_{GAKP,ELC256}$)

$P_{enc,384}$ = points(($t_{enc,1}$, $t_{enc,2}$, …, $t_{enc,256}$), $T_{enc,ELC384}$)

$P_{dec,384}$ = points(($t_{dec,1}$, $t_{dec,2}$, …, $t_{dec,256}$), $T_{dec,ELC384}$)

$P_{GAKP,384}$ = points(($t_{GAKP,1}$, $t_{GAKP,2}$, …, $t_{GAKP,256}$), $T_{GAKP,ELC384}$)

$P_{enc,512}$ = points(($t_{enc,1}$, $t_{enc,2}$, …, $t_{enc,256}$), $T_{enc,ELC512}$)

$P_{dec,512}$ = points(($t_{dec,1}$, $t_{dec,2}$, …, $t_{dec,256}$), $T_{dec,ELC512}$)

$P_{GAKP,512}$ = points(($t_{GAKP,1}$, $t_{GAKP,2}$, …, $t_{GAKP,256}$), $T_{GAKP,ELC512}$)

[<=]

Testnachbereitung:

Keine.

18.8.17 Selektieren von Ordnern und Logical Channel Reset

In diesem Abschnitt wird die Selektion eines Ordners und das Rücksetzen des Basiskanals betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.Auth.

Testvorbereitung:

G2_N217.110 - (N217.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

Testdurchführung:

G2_N217.200 - (N217.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 3 ausgeführt.**[<=]**

G2_N217.210 - (N217.210) K_Performanztest
Schritt 1: *currentFolder* MUSS gemäß (N044.900) auf / MF / DF.Auth gesetzt werden. Die Laufzeit $t_{select,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N217.220 - (N217.220) K_Performanztest
Schritt 2: GET CHALLENGE wird gemäß (N098.625) ausgeführt. Die Laufzeit $t_{rnd,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N217.230 - (N217.230) K_Performanztest
Schritt 3: Der Basiskanal MUSS gemäß Use Case aus (N099.524) zurückgesetzt werden, wobei der Kommandoparameter *logicalChannelNumber* gleich null zu setzen ist. Die Laufzeit $t_{reset,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17045 - (N217.240) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$$P_{select_DF} = \text{points}((t_{select,1}, t_{select,2}, \ldots, t_{select,100}), T_{select_DF})$$

$$P_{challenge} = \text{points}((t_{rnd,1}, t_{rnd,2}, \ldots, t_{rnd,100}), T_{challenge})$$

$$P_{reset_Ch} = \text{points}((t_{reset,1}, t_{reset,2}, \ldots, t_{reset,100}), T_{reset_Ch})$$

[<=]

Testnachbereitung:

Keine.

18.8.18 MANAGE SECURITY ENVIRONMENT

In diesem Abschnitt wird die Selektion kryptographischer Objekte betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.IAS.

Testvorbereitung:

G2_N218.110 - (N218.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N218.120 - (N218.120) K_Performanztest
Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.**[<=]**

A_17046 - (N218.130) K_Performanztest
Schritt 3: Es MUSS eine leere Menge M_{Set} = {} erstellt werden.**[<=]**

Testdurchführung:

G2_N218.200 - (N218.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.**[<=]**

G2_N218.210 - (N218.210) K_Performanztest
Schritt 1: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N100.900), wobei als *keyRef* PrK.X509_ELC256.*keyIdentifier* verwendet und *algId* = elcRoleAuthentication gesetzt wird. Die Laufzeit t_{Set} dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und der Menge M_{Set} hinzugefügt werden.**[<=]**

G2_N218.220 - (N218.220) K_Performanztest
Schritt 2: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N102.900), wobei als *keyRef* PrK.X509_ELC384.*keyIdentifier* verwendet und *algId* = signECDSA gesetzt wird. Die Laufzeit t_{Set} dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und der Menge M_{Set} hinzugefügt werden.**[<=]**

G2_N218.230 - (N218.230) K_Performanztest
Schritt 3: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N103.800), wobei als *keyRef* PrK.X509_ELC512.*keyIdentifier* verwendet und *algId* = elcSharedSecretCalculation gesetzt wird. Die Laufzeit t_{Set} dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und der Menge M_{Set} hinzugefügt werden.**[<=]**

G2_N218.240 - (N218.240) K_Performanztest
Schritt 4: MANAGE SECURITY ENVIRONMENT Restore Kommando gemäß (N099.900), wobei *seNo* = 1 gesetzt wird. Die Laufzeit $t_{restore,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden**[<=]**

Testauswertung:

A_17047 - (N218.250) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

P_{MSE_Set} = points(M_{Set} , T_{MSE_Set})

$P_{MSE_Restore}$ = points(($t_{restore,1}$, $t_{restore,2}$, …, $t_{restore,100}$), $T_{MSE_restore}$)

[<=]

Testnachbereitung:

Keine.

18.8.19 GENERAL AUTHENTICATE, PACE

Dieser Prüfpunkt ist nur relevant, wenn Option_kontaktlose_Schnittstelle vorhanden ist.

In diesem Abschnitt wird die Etablierung eines Trusted Channels mit einem symmetrischen Kartenverbindungsobjekt betrachtet. Dieser Prüfpunkt arbeitet mit dem Objekt / MF / DF.LCS / CAN_256.

Testvorbereitung:

G2_N219.110 - (N219.110) K_Performanztest, Option_kontaktlose_Schnittstelle
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N219.120 - (N219.120) K_Performanztest, Option_kontaktlose_Schnittstelle
Schritt 2: *currentFolder* MUSS auf / MF / DF.LCS gesetzt werden.**[<=]**

Testdurchführung:

G2_N219.200 - (N219.200) K_Performanztest, Option_kontaktlose_Schnittstelle
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.**[<=]**

A_17048 - (N219.210) K_Performanztest, Option_kontaktlose_Schnittstelle
Schritt 1: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N102.448) zur Auswahl von CAN_256. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N219.220 - (N219.220) K_Performanztest, Option_kontaktlose_Schnittstelle
Schritt 2: Etablierung eines vertrauenswürdigen Kanals gemäß 14.7.2.1. Die Laufzeit aller daran beteiligten Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden und zu $t_{PACE,i}$ addiert werden.**[<=]**

Testauswertung:

A_17049 - (N219.230) K_Performanztest
Der Performanztest MUSS die Performanzpunkte P_{PACE} wie folgt ermitteln:
P_{PACE} = points(($t_{PACE,1}$, $t_{PACE,2}$, …, $t_{PACE,100}$), T_{PACE})**[<=]**

Testnachbereitung:

Keine.

18.8.20 Symmetrische Sessionkeyaushandlung für Trusted Channel

Dieser Prüfpunkt ist nur relevant, wenn Option_Kryptobox vorhanden ist.

In diesem Abschnitt wird lediglich die symmetrische Aushandlung von Sessionkeys für Secure Messaging betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln TC.AES128, TC.AES192 und TC.AES256 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

G2_N220.110 - (N220.110) K_Performanztest, Option_Kryptobox
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N220.120 - (N220.120) K_Performanztest, Option_Kryptobox
Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.**[<=]**

Testdurchführung:

G2_N220.200 - (N220.200) K_Performanztest, Option_Kryptobox
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {TC.AES128, TC.AES192, TC.AES256} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 7 ausgeführt.**[<=]**

A_17050 - (N220.210) K_Performanztest, Option_Kryptobox
Schritt 1: MANAGE SECURITY ENVIRONMENT Set Kommandos gemäß (N100.400) ~~und (N101.400),~~ wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4TC gesetzt wird. Die Laufzeiten dieser Kommandos sind für diesen Prüfpunkt irrelevant.**[<=]**

A_17051 - (N220.215) K_Performanztest, Option_Kryptobox
Schritt 2: MANAGE SECURITY ENVIRONMENT Set Kommando gemäß (N101.400), wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4TC gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N220.220 - (N220.220) K_Performanztest, Option_Kryptobox
Schritt 3: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 4 bis 6 ausgeführt. Anschließend wird mit Schritt 7 fortgefahren.**[<=]**

G2_N220.230 - (N220.230) K_Performanztest, Option_Kryptobox
Schritt 4: Es wird ein INTERNAL AUTHENTICATE Kommando gemäß (N86.400) und (N086.902)a ausgeführt. Die Laufzeit $t_{IntAuth,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N220.240 - (N220.240) K_Performanztest, Option_Kryptobox
Schritt 5: Ein erfolgreiches EXTERNAL AUTHENTICATE Kommando (N083.500) und (N084.402)a wird durchgeführt. Die Laufzeit $t_{ExtAuth,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

A_17052 - (N220.250) K_Performanztest, Option_Kryptobox
Schritt 6: Die Ausführungszeiten werden wie folgt zusammengefasst:
$t_{SK,i} = t_{IntAuth,i} + t_{ExtAuth,i}$. **[<=]**

A_17053 - (N220.260) K_Performanztest, Option_Kryptobox
Schritt 7: MANAGE SECURITY ENVIRONMENT Restore Kommando gemäß (N099.900) mit *seNo* = 1, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

Testauswertung:

A_17054 - (N220.270) K_Performanztest, Option_Kryptobox
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$P_{SK4TC,AES128}$ = points(($t_{SK128,1}$, $t_{SK128,2}$, …, $t_{SK128,100}$), $T_{SK4TC,AES128}$)

$P_{SK4TC,AES192}$ = points(($t_{SK192,1}$, $t_{SK192,2}$, …, $t_{SK192,100}$), $T_{SK4TC,AES192}$)

$P_{SK4TC,AES256}$ = points(($t_{SK256,1}$, $t_{SK256,2}$, …, $t_{SK256,100}$), $T_{SK4TC,AES256}$)

($t_{SK256,1}$,

[<=]

Testnachbereitung:

Keine.

18.8.21 Sessionkeynutzung im Trusted Channel

Dieser Prüfpunkt ist nur relevant, wenn Option_Kryptobox vorhanden ist.

In diesem Abschnitt wird die Nutzung von Sessionkeys im Rahmen eines Trusted Channels betrachtet. Dieser Prüfpunkt arbeitet mit den Schlüsseln TC.AES128, TC.AES192 und TC.AES256 in der Anwendung / MF / DF.Auth.

Testvorbereitung:

G2_N221.110 - (N221.110) K_Performanztest, Option_Kryptobox
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

G2_N221.120 - (N221.120) K_Performanztest, Option_Kryptobox
Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.**[<=]**

Testdurchführung:

G2_N221.200 - (N221.200) K_Performanztest, Option_Kryptobox
Der Performanztest MUSS im Rahmen der Testdurchführung eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *keyIdentifier* für die Schlüssel aus der Menge {TC.AES128, TC.AES192, TC.AES256} annimmt. In jedem Schleifendurchlauf werden die Schritte 1 bis 9 ausgeführt.**[<=]**

A_17055 - (N221.210) K_Performanztest, Option_Kryptobox
Schritt 1: Etablierung von Sessionkeys zur Nutzung in einem Trusted Channel gemäß der Schritte (N220.210) bis (N220.240). Die Laufzeit aller dabei verwendeten Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

G2_N221.220 - (N221.220) K_Performanztest, Option_Kryptobox
Schritt 2: Die Testdurchführung MUSS eine innere Schleife 1000-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 8 ausgeführt. Anschließend wird mit Schritt 9 fortgefahren.**[<=]**

G2_N221.230 - (N221.230) K_Performanztest, Option_Kryptobox
Schritt 3: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *cmdData* = RAND(*i*) erzeugt werden.**[<=]**

G2_N221.240 - (N221.240) K_Performanztest, Option_Kryptobox
Schritt 4: Es wird ein PSO Encipher Kommando gemäß (N091.446) und (N091.650)c.1 ausgeführt, wobei als Kommandoparameter *M* der im vorherigen Schritt erzeugte Oktettstring *cmdData* verwendet wird. Die Laufzeit $t_{enc,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N221.250 - (N221.250) K_Performanztest, Option_Kryptobox
Schritt 5: Es wird ein PSO Compute Cryptographic Checksum Kommando gemäß (N087.228) und (N087.248)a durchgeführt, wobei als Kommandoparameter *data* das Chiffrat *cipher* aus der Antwortnachricht des vorherigen Kommandos verwendet wird. Die Laufzeit $t_{computeCC,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N221.260 - (N221.260) K_Performanztest, Option_Kryptobox
Schritt 6: In der *i*–ten Schleifeniteration MUSS ein Oktettstring *rspData* = RAND(*i*) erzeugt werden. Diese Daten werden so verschlüsselt, dass das Chiffrat *C* im folgenden PSO Decipher Kommando erfolgreich entschlüsselt wird. Zum Chiffrat *C* wird ein MAC *mac* so berechnet, dass dieser im folgenden PSO Verify Cryptographic Checksum erfolgreich verifiziert wird.**[<=]**

G2_N221.270 - (N221.270) K_Performanztest, Option_Kryptobox
Schritt 7: Es wird ein PSO Verify Cryptographic Checksum Kommando gemäß (N096.346) durchgeführt, wobei *inputTemplate* das Chiffrat *C* und den MAC *mac* aus dem vorherigen Schritt enthält. Die Laufzeit $t_{verifyCC,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

G2_N221.280 - (N221.280) K_Performanztest, Option_Kryptobox
Schritt 8: Es wird ein PSO Decipher Kommando gemäß (N089.845) und (N090.302)a durchgeführt, wobei das Chiffrat *C* aus Schritt 6 als Kommandoparameter verwendet wird. Die Laufzeit $t_{dec,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

A_17056 - (N221.290) K_Performanztest, Option_Kryptobox
Schritt 9: MANAGE SECURITY ENVIRONMENT Restore Kommando gemäß (N099.900) mit *seNo* = 1, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.**[<=]**

Testauswertung:

A_17057 - (N221.300) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:
Durch die Messpunkt (x, y) → (i, $t_{a,i}$) wird eine Ausgleichsgerade $y = m\,x + b$ gelegt.

$P_{compute128,b}$	= points(	(*b*, *b*)	,	$T_{compute128,b}$)
$P_{compute128,m}$	= points(	(1000 *m*, 1000 *m*)	,	$T_{compute128,b}$)
$P_{dec128,b}$	= points(	(*b*, *b*)	,	$T_{de128c,b}$)
$P_{dec128,m}$	= points(	(1000 *m*, 1000 *m*)	,	$T_{dec128,b}$)
$P_{enc128,b}$	= points(	(*b*, *b*)	,	$T_{enc128,b}$)
$P_{enc128,m}$	= points(	(1000 *m*, 1000 *m*)	,	$T_{enc128,b}$)
$P_{verify128,b}$	= points(	(*b*, *b*)	,	$T_{verify128,b}$)
$P_{verify128,m}$	= points(	(1000 *m*, 1000 *m*)	,	$T_{verify128,b}$)
$P_{compute192,b}$	= points(	(*b*, *b*)	,	$T_{compute192,b}$)

$P_{compute192,m}$ = points((1000 *m*, 1000 *m*) , $T_{compute192,b}$)

$P_{dec192,b}$ = points((*b*, *b*) , $T_{de192c,b}$)

$P_{dec192,m}$ = points((1000 *m*, 1000 *m*) , $T_{dec192,b}$)

$P_{enc192,b}$ = points((*b*, *b*) , $T_{enc192,b}$)

$P_{enc192,m}$ = points((1000 *m*, 1000 *m*) , $T_{enc192,b}$)

$P_{verify192,b}$ = points((*b*, *b*) , $T_{verify192,b}$)

$P_{verify192,m}$ = points((1000 *m*, 1000 *m*) , $T_{verify192,b}$)

$P_{compute256,b}$ = points((*b*, *b*) , $T_{compute256,b}$)

$P_{compute256,m}$ = points((1000 *m*, 1000 *m*) , $T_{compute256,b}$)

$P_{dec256,b}$ = points((*b*, *b*) , $T_{de256c,b}$)

$P_{dec256,m}$ = points((1000 *m*, 1000 *m*) , $T_{dec256,b}$)

$P_{enc256,b}$ = points((*b*, *b*) , $T_{enc256,b}$)

$P_{enc256,m}$ = points((1000 *m*, 1000 *m*) , $T_{enc256,b}$)

$P_{verify256,b}$ = points((*b*, *b*) , $T_{verify256,b}$)

$P_{verify256,m}$ = points((1000 *m*, 1000 *m*) , $T_{verify256,b}$)

[<=]

Hinweis CosH_433: Die Steigung m *der Ausgleichsgeraden gibt die Verarbeitungsrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von* P_m *aufgeführt ist, rechnet dies um in Sekunden pro kByte.*

Testnachbereitung:

Keine.

18.8.22 GET RANDOM

~~Dieser Prüfpunkt ist nur relevant, wenn Option_logische_Kanäle vorhanden ist.~~

In diesem Abschnitt wird die Erzeugung einer sicheren Zufallszahl betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF.

Testvorbereitung:

G2_N222.110 - (N222.110) K_Performanztest
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

Testdurchführung:

G2_N222.200 - (N222.200) K_Performanztest
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 256-mal durchlaufen. In jedem Schleifendurchlauf wird Schritt 1 ausgeführt.**[<=]**

G2_N222.210 - (N222.210) K_Performanztest
Schritt 1: GET RANDOM wird gemäß (N099.322) ausgeführt, wobei im *i*-ten Schleifendurchlauf *i* zufällige Oktette vom Prüfling zu generieren sind. Die Laufzeit $t_{rnd,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß 18.5.1.1 gemessen werden.**[<=]**

Testauswertung:

A_17058 - (N222.220) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:
Durch die Messpunkt (x, y) → (i, $t_{rnd,i}$) wird eine Ausgleichsgerade $y = m\,x + b$ gelegt.

$$P_{Random,b} = \text{points}(b, b), T_{Random,b}$$

$$P_{Random,m} = \text{points}(1000\,m, 1000\,m), T_{Random,m}$$

[<=]

Hinweis CosH_c99: Die Steigung m *der Ausgleichsgeraden gibt die Erzeugungsrate in Sekunden pro Byte an. Der Faktor 1000, der in der Berechnung von* P_m *aufgeführt ist, rechnet dies um in Sekunden pro kByte.*

Testnachbereitung:

Keine.

18.8.23 Öffnen und Schließen logischer Kanäle

Dieser Prüfpunkt ist nur relevant, wenn Option_logische_Kanäle vorhanden ist.

In diesem Abschnitt wird das Öffnen, Schließen und Rücksetzen logischer Kanäle betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF.

Testvorbereitung:

G2_N223.110 - (N223.110) K_Performanztest, Option_logische_Kanäle
Schritt 1: Der Prüfling MUSS gemäß 18.5.2.1 aktiviert werden.**[<=]**

A_17059 - (N223.120) K_Performanztest, Option_logische_Kanäle
Schritt 2: Es MÜSSEN folgende leere Mengen erstellt werden:
a. $M_{Open} = \{\}$,
b. $M_{Close} = \{\}$,
c. $M_{RST} = \{\}$.**[<=]**

Testdurchführung:

G2_N223.200 - (N223.200) K_Performanztest, Option_logische_Kanäle
Der Performanztest MUSS im Rahmen der Testdurchführung eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.**[<=]**

G2_N223.210 - (N223.210) K_Performanztest, Option_logische_Kanäle
Schritt 1: Zusätzlich zum Basiskanal MÜSSEN mittels MANAGE CHANNEL gemäß (N099.508) drei weitere logische Kanäle geöffnet werden. Die Laufzeit dieser drei Kommandos MUSS gemäß 18.5.1.1 gemessen und in die Menge M_{Open} eingestellt werden.**[<=]**

G2_N223.220 - (N223.220) K_Performanztest, Option_logische_Kanäle
Schritt 2: Die zusätzlich zum Basiskanal geöffneten logischen Kanäle MÜSSEN mittels MANAGE CHANNEL gemäß (N099.514) geschlossen werden, wobei die Reihenfolge, in welcher die Kanäle geschlossen werden zufällig bestimmt wird. Die Laufzeit dieser drei Kommandos MUSS gemäß 18.5.1.1 gemessen und in die Menge M_{Close} eingestellt werden.**[<=]**

G2_N223.230 - (N223.230) K_Performanztest, Option_logische_Kanäle
Schritt 3: Zusätzlich zum Basiskanal MÜSSEN mittels MANAGE CHANNEL gemäß (N099.508) drei weitere logische Kanäle geöffnet werden. Die Laufzeit dieser drei Kommandos MUSS gemäß 18.5.1.1 gemessen und in die Menge M_{Open} eingestellt werden.**[<=]**

G2_N223.240 - (N223.240) K_Performanztest, Option_logische_Kanäle
Schritt 4: Die Applikationsebene MUSS mittels MANAGE CHANNEL gemäß (N099.532) zurückgesetzt werden. Die Laufzeit dieses Kommandos MUSS gemäß 18.5.1.1 gemessen und in die Menge M_{RST} eingestellt werden.**[<=]**

Testauswertung:

A_17060 - (N223.250) K_Performanztest
Der Performanztest MUSS die hier ermittelten Laufzeiten wie folgt in Performanzpunkte umrechnen:

$$P_{Open} = \text{points}(M_{Open}, T_{Open})$$

$$P_{Close} = \text{points}(M_{Close}, T_{Close})$$

$$P_{RST} = \text{points}(M_{RST}, T_{RST})$$

[<=]

Testnachbereitung:

Keine.

18.9 Kartenkonfiguration für Performanztests (normativ)

Dieser Abschnitt beschreibt die Konfiguration des Prüflings für den Performanztest. Dabei ist es zulässig, die gesamte Konfiguration so auf mehrere Images aufzuteilen, dass in

den einzelnen Images einige Anwendungen fehlen. Wegen des Speicherbedarfes von DF.strukturiert und DF.transparent ist dies gegebenenfalls erforderlich.

18.9.1 Attribute des Objektsystems

A_17061 - (N251.100) K_Personalisierung
Das Objektsystem gemäß (N019.900) MUSS folgende Attribute enthalten:**[<=]**

G2_N251.100.a - (N251.100)a K_Personalisierung
Der Wert des Attributes *root* MUSS die Anwendung gemäß (N253.050) sein.**[<=]**

G2_N251.100.b - (N251.100)b K_Personalisierung
Der Wert des Attributes *answerToReset* MUSS gemäß (N251.200) sein.**[<=]**

G2_N251.100.c - (N251.100)c K_Personalisierung
Der Wert des Attributes *iccsn8* MUSS identisch zu den letzten acht Oktetten im *body* von / MF / EF.GDO gemäß (N253.310) sein.**[<=]**

G2_N251.100.d - (N251.100)d K_Personalisierung
Das Attribut *applicationPublicKeyList* MUSS alle Schlüssel der folgenden Menge enthalten:
{
  / MF / DF.Auth / PuK.RCA_ELC256 gemäß (N254.050),
  / MF / DF.Auth / PuK.RCA_ELC384 gemäß (N254.060),
  / MF / DF.Auth / PuK.RCA_ELC512 gemäß (N254.070)
}.**[<=]**

G2_N251.100.e - (N251.100)e K_Personalisierung
In *persistentCache* MUSS Platz für mindestens zehn CA-Schlüssel sein.**[<=]**

G2_N251.100.f - (N251.100)f K_Personalisierung
Das Attribut *pointInTime* MUSS den Wert 2012.08.22 = '010200080202' besitzen.**[<=]**

18.9.1.1 Answer To Reset

A_17062 - (N251.200) K_Personalisierung
Für das Attribut *answerToReset* MUSS gelten:**[<=]**

G2_N251.200.a - (N251.200)a K_Personalisierung
Der *answerToReset* MUSS den Vorgaben aus CosT_d32 entsprechen.

Tabelle 360: CosT_d32: ATR Codierung

Zeichen	Wert	Bedeutung
TS	'3B'	Initial Character (direct convention)
T0	'9x'	Format Character (TA1/TD1 indication, x = no. of HB)
TA1	'xx'	Interface Character (FI/DI, erlaubte Werte: siehe (N024.100))
TD1	'81'	Interface Character, (T=1, TD2 indication)
TD2	'B1'	Interface Character, (T=1, TA3/TB3/TD3 indication)
TA3	'FE'	Interface Character (IFSC coding)
TB3	'45'	Interface Character, (BWI/CWI coding)
TD3	'1F'	Interface Character, (T=15, TA4 indication)
TA4	'xx'	Interface Character (XI/UI coding)
Ti	HB	Historical Bytes (HB, imax. = 15)
TCK	XOR	Check Character (exclusive OR)

[<=]

G2_N251.200.b - (N251.200)b K_Personalisierung
Der ATR SOLL ein TC1 Byte mit dem Wert 'FF' enthalten. In diesem Fall MUSS T0 auf den Wert 'Dx' gesetzt werden.**[<=]**

G2_N251.200.c - (N251.200)c K_Personalisierung
Die Historical Bytes (sofern vorhanden) MÜSSEN gemäß [ISO/IEC 7816-4] codiert werden.**[<=]**

18.9.2 Allgemeine Festlegungen zu Attributstabellen

G2_N252.100.a - (N252.100)a K_Personalisierung
Für Zugriffsbedingungen gilt: Sofern nicht anders angegeben MUSS die Zugriffsbedingung für alle in diesem Dokument genannten Kommandos und Kommandovarianten ALWAYS sein und zwar für alle unterstützten Security Environments, alle unterstützten Schnittstellen und alle Life Cycle Status.**[<=]**

A_17063 - (N252.100)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass ein COS weitere Kommandos oder Kommandovarianten unterstützt und für diese im Rahmen des Performanztests herstellerspezifische Zugriffsbedingungen verwendet. **[<=]**

G2_N252.200.a - (N252.200)a K_Personalisierung
Alle Objekte MÜSSEN sich in SE#1 wie angegeben verwenden lassen.**[<=]**

A_17064 - (N252.200)b K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass Objekte eines Prüflings ~~KÖNNEN~~ in anderen SE verwendbar sind.**[<=]**

A_17065 - (N252.200)c K_Personalisierung

Wenn Objekte in anderen SE verwendbar sind, dann MÜSSEN sie dort dieselben Eigenschaften wie in SE#1 besitzen.**[<=]**

A_17066 - (N252.300)a K_TST
Für die funktionale Eignung MUSS es zulässig sein, dass Ordner eines Prüflings, für die in den Tabellen kein *applicationIdentifier* angegeben ist, herstellerspezifische *applicationIdentifier* zugeordnet werden.**[<=]**

G2_N252.300.b - (N252.300)b K_Personalisierung
Enthält eine Tabelle mit Ordnerattributen einen oder mehrere AID, dann MUSS sich dieser Ordner mittels aller angegebenen AID selektieren lassen.**[<=]**

G2_N252.300.c - (N252.300)c.1 K_Personalisierung
Enthält eine Tabelle mit Ordnerattributen keinen *fileIdentifier* (FID), so DARF dieser Ordner NICHT mittels eines *fileIdentifier* aus dem Intervall gemäß (N006.700) selektierbar sein, es sei denn, es handelt sich um den Ordner *root*, dessen optionaler *fileIdentifier* den Wert '3F00' besitzen MUSS.**[<=]**

A_17067 - (N252.300)c.2 K_TST
Enthält eine Tabelle mit Ordnerattributen keinen *fileIdentifier* (FID), dann MUSS es für die funktionale Eignung zulässig sein, wenn dem Ordner ein fileIdentifier außerhalb des Intervalls gemäß (N006.700) zugeordnet ist.**[<=]**

G2_N252.400 - (N252.400) K_Personalisierung
Enthält eine Tabelle mit Attributen einer Datei keinen *shortFileIdentifier*, so DARF sich dieses EF NICHT mittels *shortFileIdentifier* aus dem Intervall gemäß (N007.000) selektieren lassen.**[<=]**

18.9.3 Root, die Wurzelapplikation

G2_N253.050 - (N253.050) K_Personalisierung
Die Anwendung *root* MUSS die in CosT_470 dargestellten Attribute besitzen.

Tabelle 361: CosT_470: Attribute / MF

Attribute	Wert	Bemerkung
objectType	Ein Wert aus der Menge {Application, ADF}	
applicationIdentifier	'F000 0000 03'	
lifeCycleStatus	"Operational state (active)"	
fileIdentifier	*objektType* == Application → kein *fileIdentifier* *objectType* == ADF → *fileIdentifier* = '3F 00'	
children	• / MF / DF.Auth • / MF / DF.IAS • / MF / DF.LCS • / MF / DF.SelectEF • / MF / DF.strukturiert • / MF / DF.transparent • / MF / EF.ATR • / MF / EF.DIR • / MF / EF.GDO	

[<=]

18.9.3.1 / MF / EF.ATR

G2_N253.110 - (N253.110) K_Personalisierung

Die Datei EF.ATR MUSS die in CosT_477 dargestellten Attribute besitzen.

Tabelle 362: CosT_477: Attribute / MF / EF.ATR

Attribute	Wert	Bemerkung
objectType	transparentes Elementary File	
fileIdentifier	'2F 01'	
shortFileIdentifier	'1D'= 29	
lifeCycleStatus	"Operational state (active)"	
flagTransactionMode	False	
flagChecksum	True	
numberOfOctet	herstellerspezifisch	
positionLogicalEndOfFile	herstellerspezifisch	
body	'XX…YY'	siehe a. (N253.120), b. (N253.130) und c. (N253.140)

[<=]

Für das Attribut *body* gelten folgende Festlegungen:

G2_N253.120 - (N253.120) K_Personalisierung
Der Oktettstring *body* MUSS DER–TLV codierte Datenobjekte (DO) enthalten, welche lückenlos hintereinander konkateniert werden MÜSSEN.**[<=]**

G2_N253.130 - (N253.130) K_Personalisierung
In *body* MUSS an erster Stelle genau ein DO_BufferSize mit folgenden Eigenschaften enthalten sein:
a. Tag = 'E0'.
b. DO_Buffersize MUSS genau vier DO mit einem Tag '02' enthalten.
c. Das erste DO mit Tag '02' gibt die maximale Anzahl der Oktette in einer ungesicherten Kommando APDU an.
d. Das zweite DO mit Tag '02' gibt die maximale Anzahl der Oktette in einer ungesicherten Antwort an.
e. Das dritte DO mit Tag '02' gibt die maximale Anzahl der Oktette in einer gesicherten Kommando APDU an.
f. Das vierte DO mit Tag '02' gibt die maximale Anzahl der Oktette in einer gesicherten Antwort an.**[<=]**

A_17068 - (N253.140) K_TST

~~In body KÖNNEN~~ Für die funktionale Eignung MUSS es zulässig sein, dass in *body* weitere DER-TLV codierte Datenobjekte enthalten sind.**[<=]**

18.9.3.2 / MF / EF.DIR

Die Datei EF.DIR enthält eine Liste mit Anwendungstemplates gemäß [ISO/IEC 7816-4]. Diese Liste wird dann angepasst, wenn sich die Applikationsstruktur durch Löschen oder Anlegen von Anwendungen verändert.

G2_N253.210 - (N253.210) K_Personalisierung
Die Datei EF.DIR MUSS die in CosT_9f9 dargestellten Attribute besitzen.

Tabelle 363: CosT_9f9:Attribute / MF / EF.DIR

Attribute	Wert	Bemerkung
objectType	linear variables Elementary File	
fileIdentifier	'2F 00'	
shortFileIdentifier	'1E'= 30	
lifeCycleStatus	"Operational state (active)"	
flagTransactionMode	True	
flagChecksum	True	
maximumNumberOfRec.	10 Rekord	
maximumRecordLength	36 Oktett	
flagRecordLifeCycleStatus	False	
numberOfOctet	'00BE' Oktett = 190 Oktett	
recordList Rekord 1 Rekord 2 …	 '61 – L_{61} – { 4F – 05 – F0000000~~30~~ 50 – 00 53 – L_{53} – COS_Identifier }' '61 – L_{61} – {4F – L_{4F} – AID}' …	 *root*, siehe (N253.050) weitere Anwendungstemplates

[<=]

Hinweis CosH_4d7: Der Oktettstring COS_Identifier wird von der gematik herstellerspezifisch festgelegt.

G2_N253.220 - (N253.220) K_Personalisierung
Für jede im Objektsystem vorhandene Anwendung MUSS ein eigener Rekord in EF.DIR enthalten sein, der diese Anwendung beschreibt.**[<=]**

18.9.3.3 / MF / EF.GDO

G2_N253.310 - (N253.310) K_Personalisierung
Die Datei EF.GDO MUSS die in CosT_eed dargestellten Attribute besitzen.

Tabelle 364: CosT_eed: Attribute / MF / EF.GDO

Attribute	Wert	Bemerkung
objectType	transparentes Elementary File	
fileIdentifier	'2F 02'	
shortFileIdentifier	'02'= 2	
lifeCycleStatus	"Operational state (active)"	
flagTransactionMode	False	
flagChecksum	True	
numberOfOctet	'000C' Oktett = 12 Oktett	
positionLogicalEndOfFile	12	
body	'5A – 0A – 80276…'	wird personalisiert

[<=]

G2_N253.320 - (N253.320) K_Personalisierung
In *body* MUSS genau ein DER–TLV codiertes Datenobjekt DO_ICCSN mit folgenden Eigenschaften enthalten sein:
a. Tag = '5A' und Längenfeld = '0A'.
b. Für das Wertfeld MUSS gelten:
1. Das erste Oktett MUSS den Major Industry Identifier (MII) mit dem Wert '80' enthalten.
2. Die nächsten drei Nibble MÜSSEN den Country Code Deutschlands mit dem Wert '276'enthalten.
3. Die nächsten fünf Nibble MÜSSEN den Issuer Identifier enthalten.
4. Die restlichen fünf Oktette MÜSSEN BCD codiert eine Seriennummer enthalten.**[<=]**

18.9.4 Anwendung für Authentisierungsprotokolle, DF.Auth

G2_N254.005 - (N254.005) K_Personalisierung
Die Anwendung DF.Auth MUSS die in CosT_47f dargestellten Attribute besitzen.

Tabelle 365: CosT_47f: Attribute / MF / DF.Auth

Attribute	Wert	Bemerkung
objectType	Ordner	
applicationIdentifier	'F000 0000 04'	
fileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
children	• / MF / DF.Auth / PrK.Auth_ELC256 • / MF / DF.Auth / PrK.Auth_ELC384 • / MF / DF.Auth / PrK.Auth_ELC512 • / MF / DF.Auth / PuK.RCA_ELC256 • / MF / DF.Auth / PuK.RCA_ELC384 • / MF / DF.Auth / PuK.RCA_ELC512 • / MF / DF.Auth / SK.AES128 • / MF / DF.Auth / SK.AES192 • / MF / DF.Auth / SK.AES256 • / MF / DF.Auth / TC.AES128 • / MF / DF.Auth / TC.AES192 • / MF / DF.Auth / TC.AES256	

[<=]

18.9.4.1 / MF/ DF.Auth / PrK.Auth_ELC256

G2_N254.010 - (N254.010) K_Personalisierung
Der Schlüssel PrK.Auth_ELC256 MUSS die in CosT_010 dargestellten Attribute besitzen.

Tabelle 366: CosT_010: Attribute / MF / DF.Auth / PrK.Auth_ELC256

Attribute	Wert
objectType	privates Authentisierungsobjekt, ELC256
keyIdentifier	'11' = 17
lifeCycleStatus	"Operational state (active)"
privateKey	*domainParameter* gemäß brainpoolP256r1, *privateElcKey.d* = '3b47afc75fafcf62ea3546efbe0b8d4a9295892e19acf562556441c34f374810'
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {elcRoleAuthentication, elcSessionkey4SM}

[<=]

Hinweis CosH_5bd: Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:
'040597cf45e706318271161744c8a2df37daf696c3feb0e3244de79d3a7472d8eb 07d37f751e9494d4e9b12f9cea5a2105724f742d1e444a5553ec26da4e0556e9'.

18.9.4.2 / MF/ DF.Auth / PrK.Auth_ELC384

G2_N254.020 - (N254.020) K_Personalisierung
Der Schlüssel PrK.Auth_ELC384 MUSS die in CosT_f7f dargestellten Attribute besitzen.

Tabelle 367: CosT_f7f: Attribute / MF / DF.Auth / PrK.Auth_ELC384

Attribute	Wert
objectType	privates Authentisierungsobjekt, ELC384
keyIdentifier	'12' = 18
lifeCycleStatus	"Operational state (active)"
privateKey	*domainParameter* gemäß brainpoolP384r1, *privateElcKey.d* = '5055f4c2260871efbf7eac69119597bb582be3210df50756 29311a065d1328720f1982dc9d99ea1a4f3f4ad16c857ce8'
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {elcRoleAuthentication, elcSessionkey4SM}

[<=]

Hinweis CosH_7a3: Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:

'04808efcb3b017f52f46bafbca6ad5860d0f378ffe260edb0cbab2eff3f57c93c006958e0648e824c8fc211b9 d33d564d7

0698419b30919ce79cdee85b06cd445146f254faa5df04debb56bf45355728b3e8a0d90560b002601d74c50f78 d67aa6´.

18.9.4.3 / MF/ DF.Auth / PrK.Auth_ELC512

G2_N254.030 - (N254.030) K_Personalisierung

Der Schlüssel PrK.Auth_ELC512 MUSS die in CosT_c41 dargestellten Attribute besitzen.

Tabelle 368: CosT_c41: Attribute / MF / DF.Auth / PrK.Auth_ELC512

Attribute	Wert
objectType	privates Authentisierungsobjekt, ELC512
keyIdentifier	'13' = 19
lifeCycleStatus	"Operational state (active)"
privateKey	*domainParameter* gemäß brainpoolP512r1, *privateElcKey.d* = '4ef41d5494649f0a214dded44a77d617d41ca1f56795038c70e1b222ab6d6d70 3c61dd6a6a2fa26e79db5848dedf1208ad2180afab576b23f23b31084e03edc4'
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {elcRoleAuthentication, elcSessionkey4SM}

[<=]

Hinweis CosH_3c6: Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:
'048ef0a9023aae89e687c9ab56b1e16d9dc5bdb7b4773dc1f883c4f257e917fab a22c2f8f1c20f9c283aa8197d70c592b2db4abcace1acf26a0f83f8ccd71665b 88d5cb73df9b8ed006cd463683e00a0bbc982c0f470543d4bf7190d13c8ecd21 cc1b83db0421faa737d2fad35e0391d48ecd11b25b3584fb691e66fd8786ab78'.

18.9.4.4 / MF/ DF.Auth / PrK.Auth_RSA2048

Dieses Kapitel ist absichtlich leer.

(N254.040) Dieser Punkt ist absichtlich leer.

18.9.4.5 / MF/ DF.Auth / PuK.RCA_ELC256

G2_N254.050 - (N254.050) K_Personalisierung
Der Schlüssel PuK.RCA_ELC256 MUSS die in CosT_d98 dargestellten Attribute besitzen.

Tabelle 369: CosT_d98: Attribute / MF / DF.Auth / PuK.RCA_ELC256

Attribute	Wert
objectType	Öffentliches Signaturprüfobjekt, ELC256
keyIdentifier	'4445585858600112'
lifeCycleStatus	"Operational state (active)"
publicKey	*domainParameter* gemäß brainpoolP256r1, P = '0434dc65068c3633671828f73adeb8fbc01952e3de3511823448f15afc28ebea38 29be1ac9d4d10869ce5fa675275087603e4e9fc86ff3c127b9e94a14c670164d'
oid	ecdsa-with-SHA256
CHAT	OID_{flags} = oid_cvc_fl_ti flagList = 'FF FFFF FFFF FFFF'
expirationDate	2023.04.15: YYMMDD = '0203 0004 0105'

[<=]

Hinweis CosH_f08: Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = `'74320bea049777bc663c27cfb2c488b7a49b05af348d9186f86a7196c2f6bb43'`.

18.9.4.6 / MF/ DF.Auth / PuK.RCA_ELC384

G2_N254.060 - (N254.060) K_Personalisierung
Der Schlüssel PuK.RCA_ELC384 MUSS die in CosT_969 dargestellten Attribute besitzen.

Tabelle 370: CosT_969: Attribute / MF / DF.Auth / PuK.RCA_ELC384

Attribute	Wert
objectType	Öffentliches Signaturprüfobjekt, ELC384
keyIdentifier	'4445585858600212'
lifeCycleStatus	"Operational state (active)"
publicKey	*domainParameter* gemäß brainpoolP384r1, P = `'043e84af62d1330a9a6f04a2791e0935e7cfe01eb8f7ae8d96eef77c036c0569928e53700322a28454e647c85ea9c89c1d3051ce803c21e5c27a73bd8da121524d4679ded5144368523855e522cfe9a7d1ba31e691900cd02740c399ffcc10f260'`
oid	ecdsa-with-SHA384
CHAT	OIDflags = oid_cvc_fl_ti flagList = 'F F FFFF FFFF FFFF'
expirationDate	2023.04.15: YYMMDD = '0203 0004 0105'

[<=]

Hinweis CosH_172: Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = *`'3dbf52550829cec527f91de05fc3d70b47c38d1c8504623174c912afc7941afbe104b9b3d34a4eb0dd35452e2b67884a'`.*

18.9.4.7 / MF/ DF.Auth / PuK.RCA_ELC512

G2_N254.070 - (N254.070) K_Personalisierung
Der Schlüssel PuK.RCA_ELC512 MUSS die in CosT_1db: dargestellten Attribute besitzen.

Tabelle 371: CosT_1db: Attribute / MF / DF.Auth / PuK.RCA_ELC512

Attribute	Wert
objectType	Öffentliches Signaturprüfobjekt, ELC512
keyIdentifier	'4445585858600312'
lifeCycleStatus	"Operational state (active)"
publicKey	*domainParameter* gemäß brainpoolP512r1, P = '043b5e694c154f4dd0812a91273869d382f5c81748ef5797416369b445f60dd248 0d2400d1e7dd896b054ba615d8927fb10911e02753a22b1f6ec1ba18d0c1fdaa 905712f33003bec6a51c179006e7b98f8d1d1219db49944f05602b46ac511570 b136907081a5f93a0e000b7f335f76a7f8c18584fc7a7d11fddc96844ff8cb5b'
oid	ecdsa-with-SHA512
CHAT	OIDflags = oid_cvc_fl_ti flagList = 'F F FFFF FFFF FFFF'
expirationDate	2023.04.15: YYMMDD = '0203 0004 0105'

[<=]

Hinweis CosH_ac0: Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = *'285927d1d7469dc69c83d25380dd51bbbd39cc69cf74219a53700eb44d1d0827 130c6881ea13836d454eae6972b5702b1da75e9ba39c6050539c5173129d2d0b'.*

18.9.4.8 / MF/ DF.Auth / PuK.RCA_RSA2048

Dieses Kapitel ist absichtlich leer.

(N254.080) Dieser Punkt ist absichtlich leer.

18.9.4.9 / MF / DF.Auth / SK.AES128

G2_N254.090 - (N254.090) K_Personalisierung

Der Schlüssel SK.AES128 MUSS die in CosT_a24 dargestellten Attribute besitzen.

Tabelle 372: CosT_a24: Attribute / MF / DF.Auth / SK.AES128

Attribute	Wert	Bemerkung
objectType	symmetrisches Authentisierungsobjekt, AES128	
keyIdentifier	'02' = 2	
lifeCycleStatus	"Operational state (active)"	
encKey	'0102030405060708 090A0B0C0D0E0F10'	
macKey	'100F0E0D0C0B0A09 0807060504030201'	
numberScenario	0	
algorithmIdentifier	aesSessionkey4SM	

[<=]

18.9.4.10 / MF / DF.Auth / SK.AES192

G2_N254.100 - (N254.100) K_Personalisierung
Der Schlüssel SK.AES192 MUSS die in CosT_dc7 dargestellten Attribute besitzen.

Tabelle 373: CosT_dc7: Attribute / MF / DF.Auth / SK.AES192

Attribute	Wert
objectType	Symmetrisches Authentisierungsobjekt, AES192
keyIdentifier	'03' = 3
lifeCycleStatus	"Operational state (active)"
encKey	'0102030405060708 090A0B0C0D0E0F10 1112131415161718'
macKey	'1817161514131211 100F0E0D0C0B0A09 0807060504030201'
numberScenario	0
algorithmIdentifier	aesSessionkey4SM

[<=]

18.9.4.11 / MF / DF.Auth / SK.AES256

G2_N254.110 - (N254.110) K_Personalisierung
Der Schlüssel SK.AES256 MUSS die in CosT_f96 dargestellten Attribute besitzen.

Tabelle 374: CosT_f96: Attribute / MF / DF.Auth / SK.AES256

Attribute	Wert	Bemerkung
objectType	symmetrisches Authentisierungsobjekt, AES256	
keyIdentifier	'04' = 4	
lifeCycleStatus	"Operational state (active)"	
encKey	'0102030405060708 090A0B0C0D0E0F10 1112131415161718 191A1B1C1D1E1F20'	
macKey	'100F0E0D0C0B0A09 0807060504030201 201F1E1D1C1B1A19 1817161514131211'	
numberScenario	0	
algorithmIdentifier	aesSessionkey4SM	

[<=]

18.9.4.12 / MF / DF.Auth / TC.AES128

Dieses Objekt ist genau dann verfügbar, wenn Option_Kryptobox implementiert ist.

G2_N254.120 - (N254.120) K_Personalisierung, Option_Kryptobox
Der Schlüssel TC.AES128 MUSS die in CosT_96c dargestellten Attribute besitzen.

Tabelle 375: CosT_96c: Attribute / MF / DF.Auth / TC.AES128

Attribute	Wert	Bemerkung
objectType	symmetrisches Authentisierungsobjekt, AES128	
keyIdentifier	'05' = 5	
lifeCycleStatus	"Operational state (active)"	
encKey	'0102030405060708 090A0B0C0D0E0F10'	
macKey	'100F0E0D0C0B0A09 0807060504030201'	
algorithmIdentifier	aesSessionkey4TC	

[<=]

18.9.4.13 / MF / DF.Auth / TC.AES192

Dieses Objekt ist genau dann verfügbar, wenn Option_Kryptobox implementiert ist.

G2_N254.130 - (N254.130) K_Personalisierung, Option_Kryptobox
Der Schlüssel TC.AES192 MUSS die in CosT_bb5 dargestellten Attribute besitzen.

Tabelle 376: CosT_bb5: Attribute / MF / DF.Auth / TC.AES192

Attribute	Wert
objectType	Symmetrisches Authentisierungsobjekt, AES192
keyIdentifier	'06' = 6
lifeCycleStatus	"Operational state (active)"
encKey	'0102030405060708 090A0B0C0D0E0F10 1112131415161718'
macKey	'1817161514131211 100F0E0D0C0B0A09 0807060504030201'
algorithmIdentifier	aesSessionkey4TC

[<=]

18.9.4.14 / MF / DF.Auth / TC.AES256

Dieses Objekt ist genau dann verfügbar, wenn Option_Kryptobox implementiert ist.

G2_N254.140 - (N254.140) K_Personalisierung, Option_Kryptobox
Der Schlüssel TC.AES256 MUSS die in CosT_79e dargestellten Attribute besitzen.

Tabelle 377: CosT_79e: Attribute / MF / DF.Auth / TC.AES256

Attribute	Wert	Bemerkung
objectType	symmetrisches Authentisierungsobjekt, AES256	
keyIdentifier	'07' = 7	
lifeCycleStatus	"Operational state (active)"	
encKey	'0102030405060708 090A0B0C0D0E0F10 1112131415161718 191A1B1C1D1E1F20'	
macKey	'100F0E0D0C0B0A09 0807060504030201 201F1E1D1C1B1A19 1817161514131211'	
algorithmIdentifier	aesSessionkey4TC	

[<=]

18.9.5 Anwendung für IAS Services, DF.IAS

G2_N255.050 - (N255.050) K_Personalisierung

Die Anwendung DF.IAS MUSS die in CosT_c20 dargestellten Attribute besitzen.

Tabelle 378: CosT_c20: Attribute / MF / DF.IAS

Attribute	Wert	Bemerkung
objectType	Ordner	
applicationIdentifier	'F000 0000 05'	
fileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
children	• / MF / DF.IAS / PrK.X509_ELC256 • / MF / DF.IAS / PrK.X509_ELC384 • / MF / DF.IAS / PrK.X509_ELC512 • / MF / DF.IAS / PrK.X509_RSA2048 • / MF / DF.IAS / PrK.X509_RSA3072	

[<=]

18.9.5.1 / MF/ DF.IAS / PrK.X509_ELC256

G2_N255.110 - (N255.110) K_Personalisierung
Der Schlüssel PrK.X509_ELC256 MUSS die in CosT_9c2 dargestellten Attribute besitzen.

Tabelle 379: CosT_9c2: Attribute / MF / DF.IAS / PrK.X509_ELC256

Attribute	Wert
objectType	privates ELC Schlüsselobjekt, ELC256
keyIdentifier	'18' = 24
lifeCycleStatus	"Operational state (active)"
privateKey	*domainParameter* gemäß brainpoolP256r1, *privateElcKey.d* = `'295fcd16a620cb6fe34f7b0326e9b059c0b5229057c04ed9a7ed13a542967ee0'`
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {elcRoleAuthentication, elcSharedSecretCalculation, signECDSA}

[<=]

Hinweis CosH_307: Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:
`´04855978a5c0237c53404d6de6b9626145494feca14591dcfcbdb0385097759872 9ba33a5ae2dd510878c4edd669fbf6b55fddcbeffdbacfbec3b88e151fc78556´.`

18.9.5.2 / MF/ DF.IAS / PrK.X509_ELC384

G2_N255.120 - (N255.120) K_Personalisierung
Der Schlüssel PrK.X509_ELC384 MUSS die in CosT_ebf dargestellten Attribute besitzen.

Tabelle 380: CosT_ebf: Attribute / MF / DF.IAS / PrK.X509_ELC384

Attribute	Wert
objectType	privates ELC Schlüsselobjekt, ELC384
keyIdentifier	'19' = 25
lifeCycleStatus	"Operational state (active)"
privateKey	*domainParameter* gemäß brainpoolP384r1, *privateElcKey.d* = `'6fbeecf714222b4a6ce80f035a00d80508cce63c73f4ad77 8db108f8e556f8a9f7625cce92fc6d24a2de0d8e998bee58'`
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {elcSharedSecretCalculation, signECDSA}

[<=]

Hinweis CosH_b80: Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:
`´0435229000a37cc393cd04d8192ef25b910720a11453600a568b7f7adf3efebe984bc4893e3782aedfa9d4deeb602e7574`

02e591fa8bc4cfe150e0b7a7504616f19dba2e7b55c6c11a89b6061e0b2e831a7ccefd8b306400c2a0fe6e8bc5ebff84´.

18.9.5.3 / MF/ DF.IAS / PrK.X509_ELC512

G2_N255.130 - (N255.130) K_Personalisierung

Der Schlüssel PrK.X509_ELC512 MUSS die in CosT_2bb dargestellten Attribute besitzen.

Tabelle 381: CosT_2bb: Attribute / MF / DF.IAS / PrK.X509_ELC512

Attribute	Wert
objectType	privates ELC Schlüsselobjekt, ELC512
keyIdentifier	'1A' = 26
lifeCycleStatus	"Operational state (active)"
privateKey	*domainParameter* gemäß brainpoolP512r1, *privateElcKey.d* = ´2c24d847750f8321f875880deb583a7f4afdc8ebfd1fd6f587e6876d594bffbaf284814019156c9efbafdac25ec426b6c842e82c5e4a657fee934c21b0447810´
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {elcSharedSecretCalculation, signECDSA}

[<=]

Hinweis CosH_099: Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:
´043e4e30df5028cdc67a4aa3b95c4803a278e6ada8dd6d37ead2816fcf9507001d03929a692137e376ee60c7318b3bb219d84f1bfb4429ed32b549c7544e006f226c787e46f2d0ed40e15e2baec59b5f9bcd0dc8d189dc84c27c82aeb4de14183d95c5b2899f1530c8cdadddc75b262658513d5716725745f101953d10f2e8a544´.

18.9.5.4 / MF/ DF.IAS / PrK.X509_RSA2048

G2_N255.140 - (N255.140) K_Personalisierung

Der Schlüssel PrK.X509_RSA2048 MUSS die in CosT_cd0 dargestellten Attribute besitzen.

Tabelle 382: CosT_cd0: Attribute / MF / DF.IAS / PrK.X509_RSA2048

Attribute	Wert
objectType	privates RSA Schlüsselobjekt, RSA2048
keyIdentifier	'0B' = 11
lifeCycleStatus	"Operational state (active)"
modulusLength	2048

privateKey	'30 8204bd 02 01 00 30 0d 06 09 2a864886f70d010101 05 00 04 8204a7 30 8204a3 02 01 00 02 820101 00a20d2aa5d0241c4cadd5db90509a3030a4f24f99c6ad2604ca96fd06986345a4 8eb74f1b9c7be26e18295fd58dd1a3b6d715580ad27d2b5db93c168b8cdc261d 92e12eee344624317a20d2221cbbbcf1bd376d20126980f470ff2b48f65d686d 9cc5e0f7383dcab0881a7de198f1a7b1a2717d1c03ad86973dcefdb0425ed9bb 98344549eac891c9e25c3a02e873aafb5e2adc4afa1c526081bd1f61e75c43fb cbf135f572073d18d1a30c69d6aac6956761c02a36d74de4fc6bb96c7529528a 60628d99dea4abcdb209da139398a68f563ab3595bd882b583097a167b590932 c9db5d47df32c10877bd5fe3cf46a31826259ef1329b5b99a2c87a40e7833459 02 03 010001 02 820100 34e0c8fecc394c46b51ea883a1d97e4a1138c442b072c58a20b53e214dce6ee6 306f9e4fab333d82a13db6f8cf4b0df9d69b2f5c70acc95eced7cd9f81ba4071 bc00e0b877b43f912981d62fad629157ec5f4eab7d7691ceb1a481f24ff9d0de 9b9e723719520876ac22c0dac176713ae47fd5cc3363071d08dd2728c1af83aa 4f0209fe69356338dbdff20a82cc55b8bd0096332538cc7f2357f78aff6ef73d 78d618f46773ccba18cc0b77f1258d395aecb3ba6af3e197111b08a2244cee1b 5aa5d9cd76e50db7ebbabf61e36b9251c77b3e7e91c12bc090c4c1bbd7f82952 932f6f2a3c68f492de5d00af0be71214de359ae9a78740d21e86a4804ce5a501 02 8181 00c27b9a4daa01ef082498e84ced6d90bddd42c8682a7b359314cefedde7abb43a 098f93b75541d05b48d446c0fdc1c375fad17da4cf288dc6388c198739915edc bda3d6dc4c4df2ed74626d006f73ab8acf1c8842cdd58a217a3972fab547fa37 7fcaab37dc0dc6799228e204705e8acd5f31419715703c4dde4a87ae61311af9 02 8181 00d54f682555c9966346e198495568238a74dc1dc0d821224311070a3ffde4fc8b 4d61c497e9fcdc40652b16defc1c6c595999cafd3a0588a5ab890789cf9d3cf9 2cfbd518601719d456173ef4701b62062572347fe2785d57652110c30c530d1a f561d7c36104d0165047783bcdc7db6ea69a6ccdb05132a60c79fc1d2b0adc61 02 8181 00ad83a7a4990486a5e6390230f83823c631ad4eae1910ba38d27ce7c945d3dcc5 5718613b557695958c01d06a893c21ce960be8246dab09ea8b32fbe7b8c5e933 da825dc548d2d6e5624c1a62240db843ed0ad0cb81677e88e5ca71ba1a98f036 29eec119e25bebf33523029be14188673901f23a00a795360818c8bb1cfdb9c9 02 8180 0217c45ad16ba7e91371f52f8b01f98f4b3439aa81b45984b4ef0fbfaefb072b 4ab811d8b7b7fe653758e3e18e31ad327739960f43f977ecc03118fcd627a1c8 84137874c3c496414a12f2502da56721ce4e3f8b9daa36a83bdac63253b5a0e4

	49d20aeee4cdca48ddc2c5c03875a4868265f379552886c9b047a61c97268521 02 8180 6f99aa16030050301c99ea08e9d121ba6a3f9e0a32cd265cfc00200855f64041 6e0980ea47a196ecb5c48f69e8c58aa4cd2289758f7fcd384ee744e6b87d727f 8d4585c7f9924afd197469cf9910ba847841f9f67d024de97013e3c34c61b9f3 15d5f6b5cb8bfe7316193c37b4d9a99a59b962fd3c75245227b42b325d33ccf3'
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {rsaDecipherOaep, sign9796_2_DS2, signPSS}

[<=]

18.9.5.5 / MF/ DF.IAS / PrK.X509_RSA3072

G2_N255.150 - (N255.150) K_Personalisierung
Der Schlüssel PrK.X509_RSA3072 MUSS die in CosT_aa3 dargestellten Attribute besitzen.

Tabelle 383: CosT_aa3: Attribute / MF / DF.IAS / PrK.X509_RSA3072

Attribute	Wert
objectType	privates RSA Schlüsselobjekt, RSA3072
keyIdentifier	'0C' = 12
lifeCycleStatus	"Operational state (active)"
modulusLength	3072

privateKey	'30 8206fe 02 01 00 30 0d 06 09 2a864886f70d010101 05 00 04 8206e8 30 8206e4 02 01 00 02 820181 00c50ed1db3704e8259e07d6d4433f4c2c45d0cf28a78dcb8512ebd4b09cd2a07f 0ed22410b1188eb8e5faf29c72426624c4d73deee856179eef171199d5c09ceb b23a42b2866477e26c8ff34c8fd5a53320fd82e81abfa649b9b68bf876bd2b18 0d8886811bb6f48e64eded13286e39a74f61ff410cf1f6537ca2c7397a7a1250 7ad8b5a430ba1cd2c0c79cb550bf422ee4a75115050164c31a5784f1db433464 dbfdd8fafca3fb3e12f519eb2f92ae1c67c88e5bd4fba8f10726317dd114ff64 b644ba58b3a4cda9613219aa36d712300306b16f418c03f4ca5f66ecf640808b 98c547019d7b562a10b1f6d8ed9274a4e7159725ff5c20180434db6190235ae3 ac905e4a1981b3b488d11f78e1c586c85543be59bc2a313deb6914c14c8b4eb6 838fdf1a145805aaf026d22362fa1c2b5f7e686bf1ec35c59c15201a93c1c549 de51e79f32fe371c826fc99e2137268de8993bd0c2780123702fe62db3bbac9b 94d05b6764de631e8d1074df47a896a36fd46dc800966e3c70c09cb2de726f9d 02 03 010001 02 820181 008f37c11fd0b1f942f5c2fec8d0831d8a83c749f86326740728844bd57c74bc7e fbb790ccacd98b4928a4a5553cd9b340ffeaebb14a1ce017fbeec935041c742f fd1dc8c1d55909a47f8073b46e4939429ab48542c9c6064648fa3a6a7f69e6fb 1942e376cc3cc12b0881bfaf0fd064888bf24e40e9f6a52f724320064f548f20 fcd7a9d44631cda702c350ca9a8016cadc161aa0eb3d630d4584dc33a74705ba 5404ff8112c2df63f3053e2c272c34eca2252a09e6166d6797e99348bcffc4c9 17a402f30a61b59e870773401d11e4345892ea98c64da681089a449dceaa2f4e 4a806d08762c5084e3c040b688ddfad316889bff178541eb3a2781d37eed58fd 24ac55928ae9db7f7419e6be825638df0b87f1a104912f1f0605e718d686e186 4c7d54169d9df913cbf9ddb75a84a37640765037810f5aa74174c30ccc5aa2b9 a895cbecbefc59fcd89a070a04fdfb83e0bd4f6e4dcb540749eeb4291b553bd1 703c8f585d439930b710069a2de833fdb42070d38e7c4be6499d75699d52a781 02 81c1 00d570f346bc001fe691c060f81974d33d6c1a67b5c52b82891cb15a4faf22df33 42c3b417578cea4859a835b76f22c1ee83c50a6eca1f400b19586c82f84a2f8b 1ab6489793cf24b4d717dc65d73c7269577e298011c7cc3a21acd2c886d08cba 8ecbd522f3d552b9c4ff0cba0b495e2a0ae44661f7106721c75b6abd33d3d228 025da96a4ea8c8b77743330f4cefaac73d037b008afd2422b32be64c9b4bf69a 94e6a41fb057718a5200bbcf69819efb2b68669684ddee6c3ceb8f3afe76fde1 02 81c1 00ec59964e422745a2f7a850cfbcf666ec4948e5ce5b2cebb1fe927723b92235a9

	4fbda6cbebe158b6efcb2cc3c5e258afca945658e82c966fc742c43c2d8b3f82 4fa395a01de92c6fd125b929fc2fc493daace672cb2f4ca9405273a12ae74955 4440e4f34ea4b4bfca973f98fe3116c765c4039391ca533273bd606a5611eebe b61a743acc9525da1b12b4aa9db1bb74a6d690af77bf22105756883c35920ddd 7651079c2b80272dc199dfee2bde1bf3eb9406c8eaf32c19972f1c3064bc113d 02 81c0 264622d9c44ca16bcf27183c651435ed05bd966b59f7fb686ae4c2b5174ac18b 87a5605def2d2db5db9df643f681dd14d03c3d2ab0c3f9d003b37f81dcd430c2 58ffe6f48764d5c837e9e773bbacf33740a136ffe83053c6f8d2ecd958937d28 d702662de65b92820d794ee6edaf325b87a87796899f3f5f27489553683fcba7 b104eaa44d86a6b77f83655c91074930811866ae2a3e4a848e6c88d7a3e1734f 34f446ea07cd26670f5d625047331f371b09959bd82793e184eb18bc13757301 02 81c0 35619e931a59e85e8075219b69e0752a94fac3fcb19719bcdff490b81ece5d34 5a704945f186ba14c7602a42256594065048e49fbc00a611a4fdad78b208ab55 714b4f3614cb5f5ff9eafd414056419deb5bf5c8a39f913f00ea1e779e4b12c3 11615ed49d449216675975e1c637ac1f691ab67c14079076eab1311f93450599 4963fc307398942e08c356ceda431445aa90f7a8c9f0ff6e956b53e5d56ba17a 0d890ba0f82a3faf13f4ff20ba03b6688722adcd9f5340f89eaee33d02e37f39 02 81c1 009f060963514741b1addc446bc2b499010ad8a8893ae0c43b1acb0da212f80293 a3a3c7b62f1d082ccf97fff0b8ccd9caaee64a293e5fd53372cc7cc961158e81 ba6eee86597c80d0ff890e7b7a3f5fce623548e1018b21c762a01d0aaa2aad8c 239733fe7feb421e3caf25d347a9f45b78621ed808090e73e96acdb1da572d80 f9e111125a86da90c7e7a81ec9a0124d93d5943a8c77353dda990e85b8550733 6a73c1a43293a3698f0be81fc672d824510284108e860fc16feabce2b99666c1'
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE: *setAlgorithmIdentifier* = {rsaDecipherOaep, sign9796_2_DS2, signPSS}

[<=]

18.9.6 Anwendung für LCS Use Cases, DF.LCS

G2_N256.050 - (N256.050) K_Personalisierung

Die Anwendung DF.LCS MUSS die in CosT_44f dargestellten Attribute besitzen.

Tabelle 384: CosT_44f: Attribute / MF / DF.LCS

Attribute	Wert	Bemerkung
objectType	Ordner	
applicationIdentifier	'F000 0000 06'	
fileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
children	• / MF / DF.LCS / CAN_256 • / MF / DF.LCS / EF.LCS • / MF / DF.LCS / PIN.LCS • / MF / DF.LCS / PrK.LCS • / MF / DF.LCS / PuK.LCS • / MF / DF.LCS / SK.LCS	

[<=]

18.9.6.1 / MF/ DF.LCS / CAN_256

Dieses Objekt ist genau dann verfügbar, wenn Option_kontaktlose_Schnittstelle implementiert ist.

G2_N256.100 - (N256.100) K_Personalisierung, Option_kontaktlose_Schnittstelle
Der Schlüssel CAN_256 MUSS die in CosT_0bf dargestellten Attribute besitzen.

Tabelle 385: CosT_0bf: Attribute / MF / DF.LCS / CAN_256

Attribute	Wert	Bemerkung
objectType	symmetrisches Kartenverbindungsobjekt	
keyIdentifier	'10' = 16	
lifeCycleStatus	"Operational state (active)"	
can	123456	
algorithmIdentifier	für alle unterstützten SE: id-PACE-ECDH-GM-AES-CBC-CMAC-256	

[<=]

18.9.6.2 / MF / DF.LCS / EF.LCS

G2_N256.200 - (N256.200) K_Personalisierung
Die Datei EF.LCS MUSS die in CosT_96f dargestellten Attribute besitzen.

Tabelle 386: CosT_96f: Attribute / MF / DF.LCS / EF.LCS

Attribute	Wert	Bemerkung
objectType	transparentes Elementary File	
fileIdentifier	'EF 01'	
shortFileIdentifier	'01' = 1	
lifeCycleStatus	"Operational state (active)"	
flagTransactionMode	False	
flagChecksum	False	
numberOfOctet	'0001' = 1	
positionLogicalEndOfFile	1	
body	'00'	

[<=]

18.9.6.3 / MF / DF.LCS / PIN.LCS

G2_N256.300 - (N256.300) K_Personalisierung

Das Passwortobjekt PIN.LCS MUSS die in CosT_802 dargestellten Attribute besitzen.

Tabelle 387: CosT_802: Attribute / MF / DF.LCS / PIN.LCS

Attribute	Wert	Bemerkung
objectType	Reguläres Passwortobjekt	
pwdIdentifier	'01' = 1	
lifeCycleStatus	"Operational state (active)"	
secret	irrelevant	
minimumLength	4	
maximumLength	12	
startRetryCounter	3	
retryCounter	3	
transportStatus	Leer-PIN	
flagEnabled	True	
startSsecList	für alle unterstützten SE: *startSsec* = unendlich	
PUK	12345678	
pukUsage	10	

[<=]

18.9.6.4 / MF / DF.LCS / PrK.LCS

G2_N256.400 - (N256.400) K_Personalisierung

Der Schlüssel PrK.LCS MUSS die in CosT_cac dargestellten Attribute besitzen.

Tabelle 388: CosT_cac: Attribute / MF / DF.LCS / PrK.LCS

Attribute	Wert
objectType	privates ELC Schlüsselobjekt, ELC256
keyIdentifier	'11' = 17
lifeCycleStatus	"Operational state (active)"
privateKey	*domainParameter* gemäß brainpoolP256r1, *privateElcKey.d* = `'7625964fa8ca41673ac8c88a69fecfe18df90496c61f38d29310ccc4fc1484a1'`
keyAvailable	True
listAlgorithmIdentifier	für alle unterstützten SE *setAlgorithmIdentifier* = {elcRoleAuthentication}

[<=]

Hinweis CosH_fb8: Der zugehörige öffentliche Punkt gemäß P2OS(…) lautet:
´044814679aa37c8c05f6c673a2b6cb621574dfb0b79835d3399e1bf1336eebdf7f67dc4670fae6fa60d456b40a0539472d5f5dcc2bf4da6872a9a3a13717dc2dfa´.

18.9.6.5 / MF / DF.LCS / PuK.LCS

G2_N256.500 - (N256.500) K_Personalisierung
Der Schlüssel PuK.LCS MUSS die in CosT_942 dargestellten Attribute besitzen.

Tabelle 389: CosT_942: Attribute / MF / DF.LCS / PuK.LCS

Attribute	Wert
objectType	Öffentliches Signaturprüfobjekt, ELC256
keyIdentifier	'0102030405060708'
lifeCycleStatus	"Operational state (active)"
publicKey	*domainParameter* gemäß brainpoolP256r1, P = `´049bbfb368605b45a6701d4e5fecbd52fbd43ee19340d802fa8792cac0a352f0e65fd6dd301fa7866a322d74c1cd0b634eac8a8e6fc600e1cd3361063107d54b5b´`
oid	ecdsa-with-SHA256
CHAT	OIDflags = oid_cvc_fl_ti flagList = 'BF FFFF FFFF FFFF'
expirationDate	2015.03.18: YYMMDD = '0105 0003 0108'

[<=]

Hinweis CosH_e32: Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d = *`'715babdef8bc5828ce2ba351e4531a1d7ae8f64c1694ff9fb07e433231f678c6'`.*

18.9.6.6 / MF / DF.LCS / SK.LCS

G2_N256.600 - (N256.600) K_Personalisierung
Der Schlüssel SK.LCS MUSS die in CosT_82f dargestellten Attribute besitzen.

Tabelle 390: CosT_82f: Attribute / MF / DF.LCS / SK.LCS

Attribute	Wert	Bemerkung
objectType	symmetrisches Authentisierungsobjekt, AES128	
keyIdentifier	'12' = 18	
lifeCycleStatus	"Operational state (active)"	
encKey	'0102030405060708090A0B0C0D0E0F10'	
macKey	'100F0E0D0C0B0A090807060504030201'	
numberScenario	0	
algorithmIdentifier	aesSessionkey4SM	

[<=]

18.9.7 Anwendung für SELECT EF Use Cases, DF.SelectEF

G2_N257.050 - (N257.050) K_Personalisierung
Die Anwendung DF.SelectEF MUSS die in CosT_d4d dargestellten Attribute besitzen.

Tabelle 391: CosT_d4d: Attribute / MF / DF.SelectEF

Attribute	Wert	Bemerkung
objectType	Ordner	
applicationIdentifier	'F000 0000 07'	
fileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
children	• / MF / DF.SelectEF / EF.00 • / MF / DF.SelectEF / EF.… • / MF / DF.SelectEF / EF.99	

[<=]

18.9.7.1 / MF / DF.SelectEF / EF.xx

G2_N257.110 - (N257.110) K_Personalisierung
Die Dateien EF.xx MÜSSEN die in CosT_9d6 dargestellten Attribute besitzen.

Tabelle 392: CosT_9d6: Attribute / MF / DF.SelectEF / EF.xx

Attribute	Wert	Bemerkung
objectType	transparentes Elementary File	
fileIdentifier	'EF 00' + *i*, mit *i* = 0, 1, 2, …, 99	
shortFileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
flagTransactionMode	False	
flagChecksum	False	
numberOfOctet	'0001' = 1	
positionLogicalEndOfFile	1	
body	'00'	

[<=]

Hinweis CosH_bf6: CosT_9d6 spezifiziert 100 Dateien, welche sich lediglich im Attribut fileIdentifier *unterscheiden. Gemäß der Tabelle gilt für die Menge der* fileIdentifier: {'EF00', 'EF01', …, EF09', 'EF0A', …, EF0F', 'EF10', …, 'EF63'}*.*

18.9.8 Anwendung für rekordorientierte Dateioperationen, DF.strukturiert

G2_N258.050 - (N258.050) K_Personalisierung
Die Anwendung DF.strukturiert MUSS die in CosT_79d dargestellten Attribute besitzen.

Tabelle 393: CosT_79d: Attribute / MF / DF.strukturiert

Attribute	Wert	Bemerkung
objectType	Ordner	
applicationIdentifier	'F000 0000 08'	
fileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
children	• / MF / DF.strukturiert / EF.strukturiert	

[<=]

18.9.8.1 / MF / DF.strukturiert / EF.strukturiert

G2_N258.110 - (N258.110) K_Personalisierung

Die Datei EF.strukturiert MUSS die in CosT_f1d dargestellten Attribute besitzen.

Tabelle 394: CosT_f1d: Attribute / MF / DF.strukturiert / EF.strukturiert

Attribute	Wert	Bemerkung
objectType	linear variables Elementary File	
fileIdentifier	'EF 01'	
shortFileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
flagTransactionMode	True	
flagChecksum	True	
maximumNumberOfRec.	254	
maximumRecordLength	'FF'= 255	
flagRecordLifeCycleStatus	True	
numberOfOctet	'FD02' = 64.770	
recordList	leere Liste	

[<=]

18.9.9 Anwendung für transparente Dateioperationen, DF.transparent

G2_N259.050 - (N259.050) K_Personalisierung
Die Anwendung DF.transparent MUSS die in CosT_64f dargestellten Attribute besitzen.

Tabelle 395: CosT_64f: Attribute / MF / DF.transparent

Attribute	Wert	Bemerkung
objectType	Ordner	
applicationIdentifier	'F000 0000 09'	
fileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
children	• / MF / DF.transparent / EF.transparent	

[<=]

18.9.9.1 / MF / DF.transparent / EF.transparent

G2_N259.110 - (N259.110) K_Personalisierung

Die Datei EF.transparent MUSS die in CosT_400 dargestellten Attribute besitzen.

Tabelle 396: CosT_400: Attribute / MF / DF.transparent / EF.transparent

Attribute	Wert	Bemerkung
objectType	transparentes Elementary File	
fileIdentifier	'EF 01'	
shortFileIdentifier	-	
lifeCycleStatus	"Operational state (active)"	
flagTransactionMode	True	
flagChecksum	True	
numberOfOctet	'8000' = 32.768	
positionLogicalEndOfFile	0	
body	irrelevant	

[<=]

19 Domainparameter elliptischer Kurven (informativ)

Dieses Kapitel enthält die Domainparameter der elliptischen Kurven, die gemäß dieser Dokumentenversion vom COS zu unterstützen sind. Die Object Identifier der hier dargestellten Kurven sind CosT_a91 entnommen.

19.1 ansix9p256r1

ansix9p256r1: objectIdentifier = {1.2.840.10045.3.1.7} = '2A8648CE3D030107'

Die hier dargestellten Domainparameter wurden [ANSI X9.62#L.6.4.3] entnommen und sind identisch zu [FIPS 186–4#D.1.2.3 P-256]:

Tabelle 397: CosT_f68, Kurvenparameter ansix9p256r1

p	'FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF'
a	'FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC'
b	'5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B'
G	in komprimierter Form: '03 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296' in Koordinatenform: x_g='6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296' y_g='4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5'
n	'FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551'
h	1

Hinweis CosH_1a4: Gegenüber der Darstellung in [FIPS 186–4#D.1.2.3] ist zu beachten das gilt: -3 ≡ a mod p.

19.2 ansix9p384r1

ansix9p384r1: objectIdentifier = {1.3.132.0.34} = '2B81040022'

Die hier dargestellten Domainparameter wurden [ANSI X9.62#L.6.5.2] entnommen und sind identisch zu [FIPS 186–4#D.1.2.4 P-384]::

Tabelle 398: CosT_88a: Kurvenparameter ansix9p384r1

p	'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFF'
a	'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFC'
b	'B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112 0314088F 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF'
G	in komprimierter Form: '03 AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98 59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7' in Koordinatenform: x_g='AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98 59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7' y_g='3617DE4A 96262C6F 5D9E98BF 9292DC29 F8F41DBD 289A147C E9DA3113 B5F0B8C0 0A60B1CE 1D7E819D 7A431D7C 90EA0E5F'
n	'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973'
h	1

Hinweis CosH_9a8: Gegenüber der Darstellung in [FIPS 186–4#D.1.2.3] ist zu beachten das gilt: -3 ≡ a mod p.

19.3 brainpoolP256r1

brainpoolP256r1: objectIdentifier = {1.3.36.3.3.2.8.1.1.7} = '2B2403030208010107'

Die hier dargestellten Domainparameter wurden [RFC5639#3.4] entnommen:

Tabelle 399: CosT_de0: Kurvenparameter brainpoolP256r1

p	' A9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377'
a	' 7D5A0975FC2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9'
b	' 26DC5C6CE94A4B44F330B5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B6'
G	in Koordinatenform: x_g='8BD2AEB9CB7E57CB2C4B482FFC81B7AFB9DE27E1E3BD23C23A4453BD9ACE3262' y_g='547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E545C1D54C72F046997'
n	' A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7'
h	1

19.4 brainpoolP384r1

brainpoolP384r1: objectIdentifier = {1.3.36.3.3.2.8.1.1.11} = '2B240303020801010B'

Die hier dargestellten Domainparameter wurden [RFC5639#3.6] entnommen:

Tabelle 400: CosT_62f: Kurvenparameter brainpoolP384r1

p	'8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B4 12B1DA197FB71123ACD3A729901D1A71874700133107EC53'
a	'7BC382C63D8C150C3C72080ACE05AFA0C2BEA28E4FB22787 139165EFBA91F90F8AA5814A503AD4EB04A8C7DD22CE2826'
b	'04A8C7DD22CE28268B39B55416F0447C2FB77DE107DCD2A6 2E880EA53EEB62D57CB4390295DBC9943AB78696FA504C11'
G	in Koordinatenform: x_g='1D1C64F068CF45FFA2A63A81B7C13F6B8847A3E77EF14FE3 DB7FCAFE0CBD10E8E826E03436D646AAEF87B2E247D4AF1E' y_g='8ABE1D7520F9C2A45CB1EB8E95CFD55262B70B29FEEC5864 E19C054FF99129280E4646217791811142820341263C5315'
n	'8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B3 1F166E6CAC0425A7CF3AB6AF6B7FC3103B883202E9046565'
h	1

19.5 brainpoolP512r1

brainpoolP512r1: objectIdentifier = {1.3.36.3.3.2.8.1.1.13} = '2B240303020801010D'

Die hier dargestellten Domainparameter wurden [RFC5639#3.7] entnommen:

Tabelle 401: CosT_198: Kurvenparameter brainpoolP512r1

p	'AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA70330871 7D4D9B009BC66842AECDA12AE6A380E62881FF2F2D82C68528AA6056583A48F3'
a	'7830A3318B603B89E2327145AC234CC594CBDD8D3DF91610A83441CAEA9863BC 2DED5D5AA8253AA10A2EF1C98B9AC8B57F1117A72BF2C7B9E7C1AC4D77FC94CA'
b	'3DF91610A83441CAEA9863BC2DED5D5AA8253AA10A2EF1C98B9AC8B57F1117A7 2BF2C7B9E7C1AC4D77FC94CADC083E67984050B75EBAE5DD2809BD638016F723'
G	in Koordinatenform: x_g='81AEE4BDD82ED9645A21322E9C4C6A9385ED9F70B5D916C1B43B62EEF4D0098E FF3B1F78E2D0D48D50D1687B93B97D5F7C6D5047406A5E688B352209BCB9F822' y_g='7DDE385D566332ECC0EABFA9CF7822FDF209F70024A57B1AA000C55B881F8111 B2DCDE494A5F485E5BCA4BD88A2763AED1CA2B2FA8F0540678CD1E0F3AD80892'
n	'AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA70330870 553E5C414CA92619418661197FAC10471DB1D381085DDADDB58796829CA90069'
h	1

20 Erläuterungen zu Wertebereichen (informativ)

Hinweis CosH_32f: Der Hauptteil des Dokumentes enthält im Wesentlichen normative Festlegungen, WIE etwas zu funktionieren hat. Dieser Abschnitt beschreibt WARUM gewisse Festlegungen so und nicht anders getroffen wurden. Derartige Begründungen und Designentscheidungen belasten damit einerseits nicht den Hauptteil und andererseits werden Entscheidungen und Begründungen nachvollziehbar.

1. Wertebereich *pwdIdentifier*, siehe (N015.000): Gemäß [ISO/IEC 7816-4] wird das Attribut *pwdIdentifier* an der Schnittstelle zur Smartcard in den Bits b5 bis b1 des Parameters P2 codiert. Daraus ergibt sich ein Wertebereich von 1 bis 31 bzw. 0 bis 31, wenn der Wert "no information given" explizit dem Wert *pwdIdentifier* = 0 zugeordnet wird. In den Generationen 0 und 1 wurde lediglich der Bereich [0, 3] genutzt, um den Overhead bei der Verwaltung von Sicherheitszuständen klein zu halten. Durch Einführung von "Multireferenz-PIN" erscheint es sinnvoll, den Wertebereich, wie in (N015.000) dargestellt, zu vergrößern.
2. Wertebereich *startSsec*, siehe (N015.800):
   a. Als Infimum wurde 1 gewählt, da der Wert 0 für *startSsec* sinnlos ist.
   b. Als Supremum wurde 250 gewählt, damit für 1-Byte-Codierungen noch genügend Codierungsmöglichkeiten für "unendlich", Escape, RFU und ähnliches vorhanden sind.
3. Die Codierung für ShortPassword und LongPassword wurde absichtlich gleich gewählt, weil in [ISO/IEC 7816-4] nur ein begrenzter Vorrat an passenden Trailern vorhanden ist. Wenn es für notwendig erachtet wird auch an der Schnittstelle zwischen den Fällen ShortPassword und LongPassword zu unterscheiden, dann ist es sinnvoll den Trailer für LongPassword zu ändern.

21 Verzeichnisse (informativ)

21.1 Abkürzungen

Kürzel	Erläuterung
AID	Application Identifier, siehe [ISO/IEC 7816-5], Identifikator für eine Smartcard Applikation
APDU	Application Protocol Data Unit, siehe [ISO/IEC 7816-3], kleinste Nachrichteneinheit, die auf der Applikationsschicht mit der Smartcard ausgetauscht wird
BGT	"block guard time", siehe [ISO/IEC 7816-3#11.2]
CHAT	Certificate Holder Authorisation Template, Datenobjekt gemäß [gemSpec_PKI#6.7.2.5], Bestandteil eines CV-Zertifikates
COS	Card Operating System, Betriebssystem einer Smartcard
DF	Dedicated File, siehe [ISO/IEC 7816-4], Bezeichnung für einen Ordner innerhalb des Objektsystems einer Smartcard
EF	Elementary File, siehe [ISO/IEC 7816-4], Bezeichnung für eine Datei innerhalb des Objektsystems einer Smartcard
ELC	Elliptic Curve Cryptography, Kryptographie mittels elliptischer Kurven
G1	Abkürzung für Generation 1, bezeichnet eine frühere Version des Dokumentes, in der Regel ergänzt um den Zusatz "normativ"
G2	Abkürzung für Generation 2, bezeichnet diese Version dieses Dokumentes, in der Regel ergänzt um den Zusatz "normativ"
IC	Integrated Chip, Halbleiter
ICC	Integrated Chip Card, Smartcard
IFD	Interface Device
IFSC	maximum Information Field Size for the Card, siehe [ISO/IEC 7816-3#11.4.2]
IFSD	maximum Information Field Size for the Interface Device, [ISO/IEC 7816-3#11.4.2]
MAC	Message Authentication Code
MSBit	Most Significant Bit
MSByte	Most Significant Byte
LSBit	Least Significant Bit
LSByte	Least Significant Byte
TPDU	Transport Protocol Data Unit, siehe [ISO/IEC 7816-3], kleinste Nachrichteneinheit, die auf dem Data Link Layer mit der Smartcard ausgetauscht wird

21.2 Glossar

Begriff	Erläuterung
ephemer	Nur für kurze Zeit bestehend, flüchtig, ohne bleibende Bedeutung (siehe http://de.wiktionary.org/wiki/ephemer)
flagList	Bestandteil eines CHAT, der anzeigt, welche Zugriffsrechte durch eine erfolgreiche Authentisierung erworben werden
Nibble	Ein Nibble oder Nybble ist eine Datenmenge, die 4 Bits umfasst, es wird auch Halbbyte, Tetrade oder Quadrupel genannt (siehe http://de.wikipedia.org/wiki/Nibble).
~~schwacher DES-Schlüssel~~	~~Ein schwacher oder halbschwacher DES-Schlüssel senkt die Stärke einer Verschlüsselung. Zu weiteren Informationen und einer Liste schwacher und halbschwacher Schlüssel siehe:~~ • ~~Schneier, B., "Applied Cryptography, Protocols, Algorithms, and Source Code in C", 2nd edition.~~ • ~~RFC 2409 Appendix A~~ • ~~http://en.wikipedia.org/wiki/Weak_key~~

Das Glossar wird als eigenständiges Dokument zur Verfügung gestellt.

21.3 Abbildungsverzeichnis

Abbildung 1, CosA_e09: Message Sequence Chart für die Kommandobearbeitung 150
Abbildung 2: CosA_8fc: Zeitlicher Ablauf eines Roll-Back-Kommandos.....189
Abbildung 3: CosA_559: Kommunikationsmodell Gegenstelle, Steuersoftware und COS 446
Abbildung 4: CosA_336: Sequenzdiagramm zur externen Authentisierung.....447
Abbildung 5: CosA_e0a: Sequenzdiagramm zur internen Authentisierung.....449
Abbildung 6: CosA_a81: Sequenzdiagramm symmetrische Sessionkey-Aushandlung .450
Abbildung 7: CosA_8da: Sequenzdiagramm PACE Authentisierung.....452
~~Abbildung 8, CosA_b19: Sequenzdiagramm RSA-Sessionkey-Aushandlung~~ 454
Abbildung 9: CosA_7f8: Sequenzdiagramm ELC-Sessionkey-Aushandlung.....456
Abbildung 10: CosA_397: Sequenzdiagramm zur Übertragung von Sessionkeys.....457
Abbildung 11: CosA_a56: Komponentendiagramm Performanzmessplatz.....468
Abbildung 12: CosA_3de: Graphische Darstellung von $P_{gesamtEinfach}$.....482
Abbildung 13: CosA_88c: Synchrone Kommunikation zwischen CMS und Smartcard ..584
Abbildung 14: CosA_e47: Asynchrone Kommunikation zwischen CMS und Smartcard 584
Abbildung 15: CosA_85c: Zusammenhang Schlüsselspeicher.....593

21.4 Tabellenverzeichnis

Tabelle 1: CosT_1e0: Präfixe, die auf Vielfachen von Zehnerpotenzen beruhen 24
Tabelle 2: CosT_c5e: Präfixe, die auf Vielfachen von Zweierpotenzen beruhen 24
Tabelle 3: CosT_9f7: Abkürzungen und Definitionen 25
Tabelle 4: CosT_485: Liste der Komponenten, an welche dieses Dokument Anforderungen stellt.....27
Tabelle 5: CosT_89d: Definition der Funktion BitLength(...) 34
Tabelle 6: CosT_f2b: Definition der Funktion OctetLength(...).....35
Tabelle 7: CosT_042: Definition der Funktion I2OS(...).....36
Tabelle 8: CosT_c6a: Definition der Funktion OS2I(...) 36
Tabelle 9: CosT_3e0: Definition der Funktion OS2P(...).....37
Tabelle 10: CosT_e43: Definition der Funktion P2OS(...).....38
Tabelle 11: CosT_638: Definition der Funktion Extract_MSBit(...).....38
Tabelle 12: CosT_1ff: Definition der Funktion Extract_MSByte(...).....39
Tabelle 13: CosT_552: Definition der Funktion PaddingIso(...) 39
Tabelle 14: CosT_9fb: Definition der Funktion TruncateIso(...) 40
Tabelle 15: CosT_24c: Definition der Funktion MGF(...) 40
Tabelle 16: CosT_7d4: Definition der Funktion RAND(...) 41
Tabelle 17: CosT_c0e: Definition der Funktion ceiling(...) 42
Tabelle 18: CosT_6ed: Definition der Funktion floor(...) 42
Tabelle 19: CosT_12c: Definition der Funktion SHA_1(...) 43
Tabelle 20: CosT_159: Definition der Funktion SHA_256(...) 44
Tabelle 21: CosT_a9f: Definition der Funktion SHA_384(...) 44
Tabelle 22: CosT_676: Definition der Funktion SHA_512(...) 45
~~Tabelle 23, CosT_084: Definition der Funktion KeyDerivation_3TDES(...)~~.....45
Tabelle 24: CosT_7f9: Definition der Funktion KeyDerivation_AES128(...) 46
Tabelle 25: CosT_cae: Definition der Funktion KeyDerivation_AES192(...) 47
Tabelle 26: CosT_a21: Definition der Funktion KeyDerivation_AES256(...) 47
Tabelle 27: CosT_f06: Definition der Funktion KDF(...) 48
~~Tabelle 28, CosT_c71: Definition der Funktion DES_ENC(...)~~.....49
~~Tabelle 29, CosT_01b: Definition der Funktion 3TDES_ENC(...)~~ 49
~~Tabelle 30, CosT_a36: Definition der Funktion DES_DEC(...)~~ 50
~~Tabelle 31, CosT_478: Definition der Funktion 3TDES_DEC(...)~~ 50
Tabelle 32: CosT_ee7: Definition der Funktion AES_ENC(...).....51

Tabelle 33: CosT_bed: Definition der Funktion AES_DEC(...)....51
Tabelle 34: CosT_689: Liste der Schlüsselparameter eines RSA-Schlüssels 52
~~Tabelle 35, CosT_d3d: Definition der Funktion CALCULATE_Retail_MAC(...)~~....54
Tabelle 36: CosT_fd0: Definition der Funktion CalculateCMAC_NoPadding(...)....54
Tabelle 37: CosT_245: Definition der Funktion CalculateCMAC_IsoPadding(...) 55
~~Tabelle 38, CosT_590: Definition der Funktion VERIFY_Retail_MAC(...)~~....56
Tabelle 39: CosT_fa6: Definition der Funktion VerifyCMAC_NoPadding(...) 56
Tabelle 40: CosT_2ad: Definition der Funktion VerifyCMAC_IsoPadding(...) 57
~~Tabelle 41, CosT_533: Definition der Funktion RSA_ISO9796_2_DS1_SIGN~~....57
Tabelle 42: CosT_234: Definition der Funktion RSASSA_PKCS1_V1_5_SIGN(...) 58
Tabelle 43: CosT_858: Definition der Funktion RSA_PSS_SIGN(...) 59
Tabelle 44: CosT_ace: Definition der Funktion RSA_ISO9796_2_DS2_SIGN(...)....60
Tabelle 45: CosT_7cc: Definition der Funktion RSASSA_PSS_SIGN(...)....60
Tabelle 46: CosT_78b: Definition der Funktion ELC_SIG(...) 61
~~Tabelle 47, CosT_adc: Definition der Funktion RSA_ISO9796_2_DS1_VERIFY(...)~~....61
Tabelle 48: CosT_b2b: Definition der Funktion ELC_VER_SIG(...)....63
~~Tabelle 49, CosT_20c: Definition der Funktion 3TDES_CBX_ENC(...)~~ 64
Tabelle 50: CosT_b34: Definition der Funktion AES_CBC_ENC(...) 65
~~Tabelle 51, CosT_ef0: Definition der Funktion 3TDES_CBX_DEC(...)~~ 66
Tabelle 52: CosT_b49: Definition der Funktion AES_CBC_DEC(...) 66
Tabelle 53: CosT_089: Definition der Funktion RSAES_OAEP_ENCRYPT(...)....67
Tabelle 54: CosT_eb0: Definition der Funktion ECKApoint(...)....68
Tabelle 55: CosT_e3a: Definition der Funktion ECKAvalue(...) 69
Tabelle 56: CosT_a09: Definition der Funktion ELC_ENC(...)....69
Tabelle 57: CosT_c8a: Definition der Funktion RSAES_OAEP_DECRYPT(...)....70
Tabelle 58: CosT_bb4: Definition der Funktion ELC_DEC(...)....71
~~Tabelle 59, CosT_ad5: CV-Zertifikat einer CA mit CPI = '21', SHA-256~~ 76
~~Tabelle 60, CosT_6ea: CV-Zertifikat zur Authentisierung mit CPI = '22', SHA-256~~....77
Tabelle 61: CosT_92d: Liste der Domainparameter einer elliptischen Kurve 84
Tabelle 62: CosT_16b: Aufheben des Transportschutzes....87
Tabelle 63: CosT_65c: Definition der Funktion SearchPwd(...) 127
Tabelle 64: CosT_a08: Definition der Funktion SearchKey(...)....128
Tabelle 65: CosT_d11: Definition der Funktion SearchSecretkey(...) 129
Tabelle 66: CosT_995: Definition der Funktion SearchPublicKey(...) 130
Tabelle 67: CosT_995: Definition der Funktion StoreInCache(...)....132

Tabelle 68: CosT_6f1: Definition von Begriffen, die bei der Beschreibung von Zugriffsregeln verwendet werden 137
Tabelle 69: CosT_fe2: Definition der Funktion AccessRuleEvaluation(...) 141
Tabelle 70: CosT_760: Codierung des CLA-Bytes für Kanalnummern kleiner vier 152
Tabelle 71: CosT_67a: Codierung des CLA-Bytes für Kanalnummern größer gleich vier 152
Tabelle 72: CosT_93c: Definition von WildCard-Werten für das LeFeld 154
Tabelle 73: CosT_2be: Case 1 Kommando-APDU 156
Tabelle 74: CosT_d12: Case 1 Antwort-APDU 156
Tabelle 75: CosT_55c: Case 2 Short Kommando-APDU 156
Tabelle 76: CosT_12e: Case 2 Extended Kommando-APDU 157
Tabelle 77: CosT_e50: Case 2 Antwort-APDU 158
Tabelle 78: CosT_fbc: Case 3 Short Kommando-APDU 158
Tabelle 79: CosT_740: Case 3 Extended Kommando-APDU 159
Tabelle 80: CosT_42d: Case 3 Antwort-APDU 159
Tabelle 81: CosT_704: Case 4 Short Kommando-APDU 160
Tabelle 82: CosT_447: Case 4 Extended Kommando-APDU 161
Tabelle 83: CosT_d0a: Case 4 Antwort-APDU 161
Tabelle 84: CosT_e4d: Definition der Funktion setSecurityStatus(...) 170
Tabelle 85: CosT_489: Definition der Funktion clearSecurityStatusKey(...) für Authentisierungsschlüssel 171
Tabelle 86: CosT_956: Definition der Funktion clearSecurityStatus(...) für Ordner 172
Tabelle 87: CosT_e7c: Definition der Funktion clearSessionkeys() 173
Tabelle 88: CosT_1ba: Definition der Funktion setPasswordStatus(...) 173
Tabelle 89: CosT_f93: Definition der Funktion clearPasswordStatus(...) 174
Tabelle 90: CosT_a2e: Definition von Oktettstring zur Schlüsselableitung 175
Tabelle 91: CosT_b83: Definition der Funktion SessionkeyDerivation() 175
Tabelle 92: CosT_d73: Definitionen im Rahmen der Ableitung von Sessionkeys 175
Tabelle 93: CosT_281: Definition der Funktionsparameter zur Sicherung einer Kommando-APDU 180
Tabelle 94: CosT_4d6: Definition von Funktionsparametern zur Sicherung einer Antwort-APDU 184
Tabelle 95: CosT_c84: Kommandos alphabetisch und numerisch sortiert 186
Tabelle 96: CosT_2b6: ACTIVATE aktuelles File 191
Tabelle 97: CosT_079: *ACTIVATE* privates oder symmetrisches Schlüsselobjekt 191
Tabelle 98: CosT_ded: *ACTIVATE* öffentliches Schlüsselobjekt 192

Tabelle 99: CosT_ee1: *ACTIVATE* Passwortobjekt....192
Tabelle 100: CosT_b59: *ACTIVATE* Antwort-APDU im Erfolgsfall....193
Tabelle 101: CosT_60b: *ACTIVATE* Antwort-APDU im Fehlerfall....193
Tabelle 102: CosT_e37: *DEACTIVATE* aktuelles File....195
Tabelle 103: CosT_a2f: *DEACTIVATE* privates oder symmetrisches Schlüsselobjekt....196
Tabelle 104: CosT_ac5: *DEACTIVATE* öffentliches Schlüsselobjekt....197
Tabelle 105: CosT_e19: *DEACTIVATE* Passwortobjekt....197
Tabelle 106: CosT_294: *DEACTIVATE* Antwort-APDU im Erfolgsfall....197
Tabelle 107: CosT_93a: *DEACTIVATE* Antwort-APDU im Fehlerfall....197
Tabelle 108: CosT_799: *DELETE aktuelles File*....200
Tabelle 109: CosT_57c: *DELETE* privates oder symmetrisches Schlüsselobjekt....201
Tabelle 110: CosT_57b: *DELETE* öffentliches Schlüsselobjekt....201
Tabelle 111: CosT_4e7: *DELETE* Passwortobjekt....202
Tabelle 112: CosT_ee2: *DELETE* Antwort-APDU im Erfolgsfall....202
Tabelle 113: CosT_6c1: *DELETE* Antwort-APDU im Fehlerfall....202
Tabelle 114: CosT_7b8: *LOAD APPLICATION* mit Command Chaining....206
Tabelle 115: CosT_253: *LOAD APPLICATION* ohne Command Chaining....207
Tabelle 116: CosT_faf: *LOAD APPLICATION* Antwort-APDU im Erfolgsfall....207
Tabelle 117: CosT_df0: *LOAD APPLICATION* Antwort-APDU im Fehlerfall....207
Tabelle 118: CosT_d79: *SELECT*, kein AID, first occurrence, keine Antwortdaten....211
Tabelle 119: CosT_4c6: *SELECT*, kein AID, first occurrence, Antwortdaten mit FCP....212
Tabelle 120: CosT_b77: *SELECT*, kein AID, next occurrence, keine Antwortdaten....213
Tabelle 121: CosT_cfb: *SELECT*, kein AID, next occurrence, Antwortdaten mit FCP....213
Tabelle 122: CosT_e60: *SELECT*, AID, first occurrence, keine Antwortdaten....214
Tabelle 123: CosT_18c: SELECT, AID, first occurrence, Antwortdaten mit FCP....215
Tabelle 124: CosT_776: *SELECT*, AID, next occurrence, keine Antwortdaten....216
Tabelle 125: CosT_203: *SELECT*, AID, next occurrence, Antwortdaten mit FCP....217
Tabelle 126: CosT_28f: *SELECT*, DF oder ADF mit FID, keine Antwortdaten....217
Tabelle 127: CosT_19d: *SELECT*, DF oder ADF mit FID, Antwortdaten mit FCP....218
Tabelle 128: CosT_0e5: *SELECT*, höhere Ebene keine Antwortdaten....219
Tabelle 129: CosT_e82: *SELECT*, höhere Ebene, Antwortdaten mit FCP....219
Tabelle 130: CosT_e0d: *SELECT*, EF mit FID, keine Antwortdaten....220
Tabelle 131: CosT_55b: SELECT, EF mit FID, Antwortdaten mit FCP....221
Tabelle 132: CosT_812: SELECT, AID, first occurrence, Antwortdaten mit FCI....222
Tabelle 133: CosT_59c: *SELECT*, Kommandoparameter im Überblick....222

Tabelle 134: CosT_906: *SELECT* Antwort-APDU im Erfolgsfall 223
Tabelle 135: CosT_0dc: *SELECT* Antwort-APDU im Fehlerfall 223
Tabelle 136: CosT_e2e: Innerhalb der Beschreibung des *SELECT*-Kommandos verwendete Definitionen 223
Tabelle 137: CosT_6fa: *TERMINATE CARD USAGE*....227
Tabelle 138: CosT_fc1: *TERMINATE CARD USAGE* Antwort-APDU im Erfolgsfall....228
Tabelle 139: CosT_d53: *TERMINATE CARD USAGE* Antwort-APDU im Fehlerfall 228
Tabelle 140: CosT_f82: *TERMINATE DF* aktueller Ordner....229
Tabelle 141: CosT_8b0: *TERMINATE DF* Antwort-APDU im Erfolgsfall....230
Tabelle 142: CosT_f24: *TERMINATE DF* Antwort-APDU im Fehlerfall....230
Tabelle 143: CosT_847: *TERMINATE* aktuelle Datei 231
Tabelle 144: CosT_1cd: *TERMINATE* privates oder symmetrisches Schlüsselobjekt 232
Tabelle 145: CosT_26d: *TERMINATE* öffentliches Schlüsselobjekt 233
Tabelle 146: CosT_770: *TERMINATE* Passwortobjekt....233
Tabelle 147: CosT_722: *TERMINATE* Antwort-APDU im Erfolgsfall....233
Tabelle 148: CosT_6af: *TERMINATE* Antwort-APDU im Fehlerfall....234
Tabelle 149: CosT_238: *ERASE BINARY*, ohne shortFileIdentifier....236
Tabelle 150: CosT_3f3: *ERASE BINARY*, mit shortFileIdentifier....237
Tabelle 151: CosT_adb: *ERASE BINARY* Antwort-APDU im Erfolgsfall 237
Tabelle 152: CosT_4c8: *ERASE BINARY* Antwort-APDU im Fehlerfall....237
Tabelle 153: CosT_a50: *READ BINARY* ohne shortFileIdentifier 240
Tabelle 154: CosT_301: *READ BINARY* mit shortFileIdentifier 241
Tabelle 155: CosT_1a8: *READ BINARY* Antwort-APDU im Erfolgsfall....241
Tabelle 156: CosT_1b0: *READ BINARY* Antwort-APDU im Fehlerfall....241
Tabelle 157: CosT_be2: SET LOGICAL EOF ohne shortFileIdentifier....244
Tabelle 158: CosT_164: SET LOGICAL EOF mit shortFileIdentifier 245
Tabelle 159: CosT_77c: SET LOGICAL EOF Antwort-APDU im Erfolgsfall....245
Tabelle 160: CosT_a26: SET LOGICAL EOF Antwort-APDU im Fehlerfall 245
Tabelle 161: CosT_cdb: UPDATE BINARY ohne shortFileIdentifier....249
Tabelle 162: CosT_e25: UPDATE BINARY mit shortFileIdentifier....249
Tabelle 163: CosT_94b: UPDATE BINARY Antwort-APDU im Erfolgsfall 249
Tabelle 164: CosT_c75: UPDATE BINARY Antwort-APDU im Fehlerfall 250
Tabelle 165: CosT_17f: WRITE BINARY ohne shortFileIdentifier....253
Tabelle 166: CosT_415: WRITE BINARY mit shortFileIdentifier....253
Tabelle 167: CosT_e5e: WRITE BINARY Antwort-APDU im Erfolgsfall....254

Tabelle 168: CosT_507: WRITE BINARY Antwort-APDU im Fehlerfall 254
Tabelle 169: CosT_498: ACTIVATE RECORD, ein Rekord, ohne shortFileIdentifier 257
Tabelle 170: CosT_5ce: ACTIVATE RECORD, ein Rekord, mit shortFileIdentifier 258
Tabelle 171: CosT_ff2: ACTIVATE RECORD, alle Rekords ab P1, ohne shortFileIdentifier 258
Tabelle 172: CosT_7b6: ACTIVATE RECORD, alle Rekords ab P1, mit shortFileIdentifier 259
Tabelle 173: CosT_066: ACTIVATE RECORD Antwort-APDU im Erfolgsfall 259
Tabelle 174: CosT_180: ACTIVATE RECORD Antwort-APDU im Fehlerfall 259
Tabelle 175: CosT_74c: APPEND RECORD, ohne shortFileIdentifier 262
Tabelle 176: CosT_057: APPEND RECORD, mit shortFileIdentifier 263
Tabelle 177: CosT_71a: APPEND RECORD Antwort-APDU im Erfolgsfall 263
Tabelle 178: CosT_8cd: APPEND RECORD Antwort-APDU im Fehlerfall 263
Tabelle 179: CosT_7b1: DEACTIVATE RECORD, ein Rekord, ohne shortFileIdentifier 268
Tabelle 180: CosT_869: DEACTIVATE RECORD, ein Rekord, mit shortFileIdentifier 268
Tabelle 181: CosT_f8e: DEACTIVATE RECORD, alle Rekords ab P1, ohne shortFileIdentifier 269
Tabelle 182: CosT_9ec: DEACTIVATE RECORD, alle Rekords ab P1, mit shortFileIdentifier 269
Tabelle 183: CosT_453: DEACTIVATE RECORD Antwort-APDU im Erfolgsfall 270
Tabelle 184: CosT_563: DEACTIVATE RECORD Antwort-APDU im Fehlerfall 270
Tabelle 185: CosT_be4: DELETE RECORD, ohne shortFileIdentifier 273
Tabelle 186: CosT_157: DELETE RECORD, mit shortFileIdentifier 273
Tabelle 187: CosT_fcd: DELETE RECORD Antwort-APDU im Erfolgsfall 274
Tabelle 188: CosT_91a: DELETE RECORD Antwort-APDU im Fehlerfall 274
Tabelle 189: CosT_e66: ERASE RECORD, ohne shortFileIdentifier 277
Tabelle 190: CosT_124: ERASE RECORD, mit shortFileIdentifier 277
Tabelle 191: CosT_e14: ERASE RECORD Antwort-APDU im Erfolgsfall 277
Tabelle 192: CosT_173: ERASE RECORD Antwort-APDU im Fehlerfall 278
Tabelle 193: CosT_dd7: READ RECORD ohne shortFileIdentifier 281
Tabelle 194: CosT_b99: READ RECORD mit shortFileIdentifier 281
Tabelle 195: CosT_bb0: READ RECORD Antwort-APDU im Erfolgsfall 281
Tabelle 196: CosT_b05: READ RECORD Antwort-APDU im Fehlerfall 282
Tabelle 197: CosT_597: SEARCH RECORD ohne shortFileIdentifier 285
Tabelle 198: CosT_771: SEARCH RECORD mit shortFileIdentifier 286
Tabelle 199: CosT_13d: SEARCH RECORD Antwort-APDU im Erfolgsfall 286

Tabelle 200: CosT_8b8: SEARCH RECORD Antwort-APDU im Fehlerfall 286
Tabelle 201: CosT_dc9: UPDATE RECORD, ohne shortFileIdentifier 289
Tabelle 202: CosT_df8: UPDATE RECORD mit shortFileIdentifier 290
Tabelle 203: CosT_d43: UPDATE RECORD Antwort-APDU im Erfolgsfall 290
Tabelle 204: CosT_aff: UPDATE RECORD Antwort-APDU im Fehlerfall 290
Tabelle 205: CosT_204: CHANGE REFERENCE DATA mit altem und neuem Benutzergeheimnis 295
Tabelle 206: CosT_95c: CHANGE REFERENCE DATA, nur neues Benutzergeheimnis 296
Tabelle 207: CosT_24b: CHANGE REFERENCE DATA Antwort-APDU im Erfolgsfall ...296
Tabelle 208: CosT_1d0: CHANGE REFERENCE DATA Antwort-APDU im Fehlerfall 296
Tabelle 209: CosT_b4f: DISABLE VERIFICATION REQUIREMENT mit Verifikationsdaten 299
Tabelle 210: CosT_a7f: DISABLE VERIFICATION REQUIREMENT ohne Verifikationsdaten 300
Tabelle 211: CosT_b71: DISABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall 300
Tabelle 212: CosT_255: DISABLE VERIFICATION REQUIREMENT Antwort-APDU im Fehlerfall 300
Tabelle 213: CosT_887: ENABLE VERIFICATION REQUIREMENT mit Verifikationsdaten 303
Tabelle 214: CosT_44c: ENABLE VERIFICATION REQUIREMENT ohne Verifikationsdaten 304
Tabelle 215: CosT_2a1: ENABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall 304
Tabelle 216: CosT_903: ENABLE VERIFICATION REQUIREMENT Antwort-APDU im Fehlerfall 304
Tabelle 217: CosT_891: GET PIN STATUS 307
Tabelle 218: CosT_a97: GET PIN STATUS Antwort-APDU im Erfolgsfall 307
Tabelle 219: CosT_973: GET PIN STATUS Antwort-APDU im Fehlerfall 308
Tabelle 220: CosT_0d1: RESET RETRY COUNTER, mit PUK, mit newSecret 310
Tabelle 221: CosT_f19: RESET RETRY COUNTER, mit PUK, ohne newSecret 311
Tabelle 222: CosT_a37: RESET RETRY COUNTER, ohne PUK, mit newSecret 312
Tabelle 223: CosT_551: RESET RETRY COUNTER, ohne PUK, ohne newSecret 312
Tabelle 224: CosT_f92: RESET RETRY COUNTER Antwort-APDU im Erfolgsfall 312
Tabelle 225: CosT_a92: RESET RETRY COUNTER Antwort-APDU im Fehlerfall 312
Tabelle 226: CosT_3b5: VERIFY, Vergleich eines Benutzergeheimnisses 316
Tabelle 227: CosT_148: VERIFY Antwort-APDU im Erfolgsfall 316
Tabelle 228: CosT_49d: VERIFY Antwort-APDU im Fehlerfall 316

Tabelle 229: CosT_c1c: EXTERNAL AUTHENTICATE ohne Antwortdaten 319
Tabelle 230: CosT_fe0: MUTUAL AUTHENTICATE mit Antwortdaten 320
Tabelle 231: CosT_1fd: EXTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall 320
Tabelle 232: CosT_e88: EXTERNAL AUTHENTICATE Antwort-APDU im Fehlerfall 320
Tabelle 233: CosT_a9a: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 1a 329
Tabelle 234: CosT_19a: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 2a 329
Tabelle 235: CosT_29a: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 3a 330
Tabelle 236: CosT_152: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 4a 331
Tabelle 237: CosT_d3c: GENERAL AUTHENTICATE gegenseitige ELC-Authentisierung, Schritt 1 332
Tabelle 238: CosT_aca: GENERAL AUTHENTICATE gegenseitige ELC-Authentisierung, Schritt 2 332
Tabelle 239: CosT_d01: GENERAL AUTHENTICATE, asynchrone, symmetrische Administration, Schritt 1 333
Tabelle 240: CosT_0d9: GENERAL AUTHENTICATE, asynchrone, symmetrische Administration, Schritt 2 334
Tabelle 241: CosT_441: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 1b 335
Tabelle 242: CosT_b7a: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 2b 335
Tabelle 243: CosT_8b7: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 3b 336
Tabelle 244: CosT_d4f: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 4b 337
Tabelle 245: CosT_8fe: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 5b 337
Tabelle 246: CosT_45a: GENERAL AUTHENTICATE, asynchrone, asymmetrische Administration, Schritt 2 338
Tabelle 247: CosT_2e0: GENERAL AUTHENTICATE Antwort-APDU im Erfolgsfall 338
Tabelle 248: CosT_b4e: GENERAL AUTHENTICATE Antwort-APDU im Fehlerfall 339
Tabelle 249, CosT_0f6: GET SECURITY STATUS KEY, symmetrischer Schlüssel 353
Tabelle 250, CosT_0b6: GET SECURITY STATUS KEY, Rollenkennung 353
Tabelle 251: CosT_3cf: GET SECURITY STATUS KEY, bitSecurityList 354
Tabelle 252: CosT_380: GET SECURITY STATUS KEY Antwort-APDU im Erfolgsfall .. 354
~~Tabelle~~ 253~~, CosT_599: GET SECURITY STATUS KEY Antwort-APDU im Fehlerfall~~ ... 354

Tabelle 254: CosT_1d2: INTERNAL AUTHENTICATE 357
Tabelle 255: CosT_710: INTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall 357
Tabelle 256: CosT_235: INTERNAL AUTHENTICATE Antwort-APDU im Fehlerfall 358
Tabelle 257: CosT_c98: Tabelle aller PERFORM SECURITY OPERATION Kommando Header 362
Tabelle 258: CosT_861: PSO Compute Cryptographic Checksum, Berechnen eines MAC 364
Tabelle 259: CosT_7e7: PSO Compute Cryptographic Checksum Antwort-APDU im Erfolgsfall 364
Tabelle 260: CosT_532: PSO Compute Cryptographic Checksum Antwort-APDU im Fehlerfall 364
Tabelle 261: CosT_c48: PSO Compute Digital Signature, Signieren ohne "message recovery" 367
Tabelle 262: CosT_f84: PSO Compute Digital Signature, Signieren "message recovery" . 368
Tabelle 263: CosT_562: PSO Compute Digital Signature Antwort-APDU im Erfolgsfall... 368
Tabelle 264: CosT_7c1: PSO Compute Digital Signature Antwort-APDU im Fehlerfall . 368
Tabelle 265: CosT_643: PSO Decipher, Entschlüsseln mittels RSA 371
Tabelle 266: CosT_767: PSO Decipher, Entschlüsseln mittels elliptischer Kurven..... 372
Tabelle 267: CosT_0f8: PSO Decipher, Entschlüsseln mittels symmetrischem Schlüssel 372
Tabelle 268: CosT_add: PSO Decipher Antwort-APDU im Erfolgsfall 373
Tabelle 269: CosT_dbf: PSO Decipher Antwort-APDU im Fehlerfall 373
Tabelle 270: CosT_da1: PSO Encipher, Verschlüsseln mittels übergebenem RSA-Schlüssel 377
Tabelle 271: CosT_b10: PSO Encipher, Verschlüsseln mittels übergebenem ELC-Schlüssel 378
Tabelle 272: CosT_c3c: PSO Encipher, Verschlüsseln mittels gespeichertem RSA-Schlüssel 379
Tabelle 273: CosT_94d: PSO Encipher, Verschlüsseln mittels gespeichertem ELC-Schlüssel 379
Tabelle 274: CosT_acb: PSO Encipher, Verschlüsseln mittels symmetrischem Schlüssel 380
Tabelle 275: CosT_6f0: PSO Encipher Antwort-APDU im Erfolgsfall 380
Tabelle 276: CosT_6c0: PSO Encipher Antwort-APDU im Fehlerfall..... 381
Tabelle 277: CosT_f40: PSO Transcipher, Umschlüsseln mittels RSA-Schlüssel..... 385
Tabelle 278: CosT_632: PSO Transcipher, Umschlüsseln von RSA nach ELC 386
Tabelle 279: CosT_c68: PSO Transcipher, Umschlüsseln mittels ELC-Schlüssel 388
Tabelle 280: CosT_8eb: PSO Transcipher, Umschlüsseln von ELC nach RSA 389
Tabelle 281: CosT_dba: PSO Transcipher Antwort-APDU im Erfolgsfall 389

Tabelle 282: CosT_2dd: PSO Transcipher Antwort-APDU im Fehlerfall 389
~~Tabelle 283, CosT_eb9: PSO Verify Certificate für RSA-Schlüssel~~ 393
Tabelle 284: CosT_952: Kombination von Kurvenparametern in CV-Zertifikaten 394
Tabelle 285: CosT_8a7: PSO Verify Certificate für ELC-Schlüssel 395
Tabelle 286, CosT_b8a: PSO Verify Certificate Antwort-APDU im Erfolgsfall 395
Tabelle 287: CosT_527: PSO Verify Certificate Antwort-APDU im Fehlerfall 395
Tabelle 288: CosT_674: PSO Verify Cryptographic Checksum, Prüfen MAC 401
Tabelle 289: CosT_5e2: PSO Verify Cryptographic Checksum Antwort-APDU im Erfolgsfall 402
Tabelle 290: CosT_804: PSO Verify Cryptographic Checksum Antwort-APDU im Fehlerfall 402
Tabelle 291: CosT_f4d: PSO Verify Digital Signature mittels übergebenem ELC-Schlüssel 405
Tabelle 292: CosT_c29: PSO Verify Digital Signature Antwort-APDU im Erfolgsfall 405
Tabelle 293: CosT_3f2: PSO Verify Digital Signature Antwort-APDU im Fehlerfall 405
Tabelle 294: CosT_b85: FINGERPRINT über das COS 407
Tabelle 295: CosT_bee: FINGERPRINT Antwort-APDU im Erfolgsfall 407
Tabelle 296: CosT_105: FINGERPRINT Antwort-APDU im Fehlerfall 407
Tabelle 297: CosT_d52: GAKP, ohne Überschreiben, ohne Referenz, ohne Ausgabe 409
Tabelle 298: CosT_39e: GAKP, ohne Überschreiben, mit Schlüsselreferenz, ohne Ausgabe 409
Tabelle 299: CosT_8ee: GAKP, ggf. Überschreiben, ohne Referenz, ohne Ausgabe 410
Tabelle 300: CosT_e3e: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, ohne Ausgabe 410
Tabelle 301: CosT_36f: GAKP, Auslesen vorhandener Schlüssel ohne Schlüsselreferenz 411
Tabelle 302: CosT_f6d: GAKP, Auslesen vorhandener Schlüssel mit Schlüsselreferenz 412
Tabelle 303: CosT_fb7: GAKP, ohne Überschreiben, ohne Schlüsselreferenz, mit Ausgabe 412
Tabelle 304: CosT_c33: GAKP, ohne Überschreiben, mit Schlüsselreferenz, mit Ausgabe 413
Tabelle 305: CosT_5d8: GAKP, ggf. Überschreiben, ohne Schlüsselreferenz, mit Ausgabe 414
Tabelle 306: CosT_cd2: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, mit Ausgabe 415
Tabelle 307: CosT_246: GENERATE ASYMMETRIC KEY PAIR, Kommandoparameter im Überblick 415

Tabelle 308: CosT_a45: GENERATE ASYMMETRIC KEY PAIR Antwort-APDU im Erfolgsfall 415
Tabelle 309: CosT_6d3: GENERATE ASYMMETRIC KEY PAIR Antwort-APDU im Fehlerfall 416
Tabelle 310, CosT_31c: GET CHALLENGE für DES oder RSA Authentisierung 419
Tabelle 311: CosT_b6b: GET CHALLENGE für AES oder ELC Authentisierung 420
Tabelle 312: CosT_652: GET CHALLENGE Antwort-APDU im Erfolgsfall 420
Tabelle 313: CosT_d95: GET CHALLENGE Antwort-APDU im Fehlerfall 420
Tabelle 314: CosT_222: GET RANDOM 421
Tabelle 315: CosT_086: GET RANDOM Antwort-APDU im Erfolgsfall 421
Tabelle 316: CosT_211: GET RANDOM Antwort-APDU im Fehlerfall 422
Tabelle 317: CosT_662: LIST PUBLIC KEY mit allen Arten öffentlicher Schlüsselobjekte 423
Tabelle 318: CosT_53a: LIST PUBLIC KEY Antwort-APDU im Erfolgsfall 423
Tabelle 319: CosT_985: LIST PUBLIC KEY Antwort-APDU im Fehlerfall 424
Tabelle 320: CosT_42a: MANAGE CHANNEL zum Öffnen eines logischen Kanals 427
Tabelle 321: CosT_5ca: MANAGE CHANNEL zum Schließen eines logischen Kanals 427
Tabelle 322: CosT_442: MANAGE CHANNEL zum Zurücksetzen eines logischen Kanals 428
Tabelle 323: CosT_901: MANAGE CHANNEL zum logischen Reset der Applikationsebene 428
Tabelle 324: CosT_558: MANAGE CHANNEL Antwort-APDU im Erfolgsfall 428
Tabelle 325: CosT_ea3: MANAGE CHANNEL Antwort-APDU im Fehlerfall 429
Tabelle 326: CosT_6fc: MSE, Restore Variante 431
Tabelle 327: CosT_1b4: MSE, Selektion symmetrischer INTERNAL AUTHENTICATE-Schlüssel 432
Tabelle 328: CosT_84e: MSE, Selektion privater INTERNAL AUTHENTICATE-Schlüssel 433
Tabelle 329: CosT_a68: MSE, Selektion symmetrischer EXTERNAL AUTHENTICATE-Schlüssel 434
Tabelle 330: CosT_b61: MSE, Selektion öffentlicher EXTERNAL AUTHENTICATE-Schlüssel 435
Tabelle 331: CosT_4f0: MSE, Selektion symmetrischer MUTUAL AUTHENTICATE-Schlüssel 436
Tabelle 332: CosT_a40: MSE, Selektion symmetrisches Kartenverbindungsobjekt ohne Kurvenangabe 437
Tabelle 333: CosT_3ce: Zuordnung von idDomainParameter zu Domainparameter 438
Tabelle 334: CosT_100: MSE, Selektion symmetrisches Kartenverbindungsobjekt mit Kurvenangabe 438

Tabelle 335: CosT_d68: MSE, Selektion privater Signaturschlüssel....439
Tabelle 336: CosT_25a: MSE, Selektion öffentlicher Zertifikatsprüfschlüssel....440
Tabelle 337: CosT_9df: MSE, Schlüsselselektion zur Ent- und Umschlüsselung....441
Tabelle 338: CosT_d68: MSE, Selektion Verschlüsselungsschlüssel....442
Tabelle 339: CosT_33b: MSE Antwort-APDU im Erfolgsfall....442
Tabelle 340: CosT_03c: MSE Antwort-APDU im Fehlerfall....442
Tabelle 341: CosT_2a4: Generische AlgorithmIdentifier für Authentisierungszwecke....459
Tabelle 342: CosT_7a2: Konkrete AlgorithmIdentifier für Authentisierungszwecke....459
Tabelle 343: CosT_605: AlgorithmIdentifier für Ver- und Entschlüsselung....460
Tabelle 344: CosT_c40: AlgorithmIdentifier für Integrität und Authentizität....460
Tabelle 345: CosT_a91: Object Identifier, alphabetisch sortiert (informativ)....461
Tabelle 346: CosT_7aa: Trailer →·Fehlername....463
Tabelle 347: CosT_719: Bedeutung PPS1 gemäß [ISO/IEC 7816-3#Table 7 und 8]....469
Tabelle 348: CosT_508: Definition der Funktion P = points(...)....473
Tabelle 349: CosT_1b9: Gesamtbewertung für das Basisbetriebssystem....474
Tabelle 350: CosT_930: Gesamtbewertung für Option_kontaktlose_Schnittstelle....478
Tabelle 351: CosT_005: Gesamtbewertung für Option_Kryptobox....479
Tabelle 352: CosT_51e: Gesamtbewertung für Option_logische_Kanäle....480
Tabelle 353: CosT_439: Gesamtbewertung je nach Kombination der Optionen....481
Tabelle 354: CosT_4ba: Character Guard Time (CGT) gemäß [ISO/IEC 7816-3#11.2] 482
Tabelle 355: CosT_3d5: Performanzpunkte für LCS Operationen....489
Tabelle 356: CosT_ece: ELC-256 Sub-CA CV-Zertifikate für den Performanztest....501
Tabelle 357: CosT_811: ELC-384 Sub-CA CV-Zertifikate für den Performanztest....503
Tabelle 358: CosT_649: ELC-384 Sub-CA CV-Zertifikate für den Performanztest....506
Tabelle 359: CosT_b22: Definition der Funktion PerformanceAuth(...)....509
Tabelle 360: CosT_d32: ATR Codierung....530
Tabelle 361: CosT_470: Attribute / MF....532
Tabelle 362: CosT_477: Attribute / MF / EF.ATR....533
Tabelle 363: CosT_9f9:Attribute / MF / EF.DIR....534
Tabelle 364: CosT_eed: Attribute / MF / EF.GDO....535
Tabelle 365: CosT_47f: Attribute / MF / DF.Auth....536
Tabelle 366: CosT_010: Attribute / MF / DF.Auth / PrK.Auth_ELC256....537
Tabelle 367: CosT_f7f: Attribute / MF / DF.Auth / PrK.Auth_ELC384....537
Tabelle 368: CosT_c41: Attribute / MF / DF.Auth / PrK.Auth_ELC512....538
Tabelle 369: CosT_d98: Attribute / MF / DF.Auth / PuK.RCA_ELC256....538

Tabelle 370: CosT_969: Attribute / MF / DF.Auth / PuK.RCA_ELC384 539
Tabelle 371: CosT_1db: Attribute / MF / DF.Auth / PuK.RCA_ELC512 540
Tabelle 372: CosT_a24: Attribute / MF / DF.Auth / SK.AES128 541
Tabelle 373: CosT_dc7: Attribute / MF / DF.Auth / SK.AES192 541
Tabelle 374: CosT_f96: Attribute / MF / DF.Auth / SK.AES256 542
Tabelle 375: CosT_96c: Attribute / MF / DF.Auth / TC.AES128 542
Tabelle 376: CosT_bb5: Attribute / MF / DF.Auth / TC.AES192 543
Tabelle 377: CosT_79e: Attribute / MF / DF.Auth / TC.AES256 544
Tabelle 378: CosT_c20: Attribute / MF / DF.IAS 544
Tabelle 379: CosT_9c2: Attribute / MF / DF.IAS / PrK.X509_ELC256 545
Tabelle 380: CosT_ebf: Attribute / MF / DF.IAS / PrK.X509_ELC384 545
Tabelle 381: CosT_2bb: Attribute / MF / DF.IAS / PrK.X509_ELC512 546
Tabelle 382: CosT_cd0: Attribute / MF / DF.IAS / PrK.X509_RSA2048 546
Tabelle 383: CosT_aa3: Attribute / MF / DF.IAS / PrK.X509_RSA3072 548
Tabelle 384: CosT_44f: Attribute / MF / DF.LCS 551
Tabelle 385: CosT_0bf: Attribute / MF / DF.LCS / CAN_256 551
Tabelle 386: CosT_96f: Attribute / MF / DF.LCS / EF.LCS 552
Tabelle 387: CosT_802: Attribute / MF / DF.LCS / PIN.LCS 553
Tabelle 388: CosT_cac: Attribute / MF / DF.LCS / PrK.LCS 554
Tabelle 389: CosT_942: Attribute / MF / DF.LCS / PuK.LCS 554
Tabelle 390: CosT_82f: Attribute / MF / DF.LCS / SK.LCS 555
Tabelle 391: CosT_d4d: Attribute / MF / DF.SelectEF 555
Tabelle 392: CosT_9d6: Attribute / MF / DF.SelectEF / EF.xx 556
Tabelle 393: CosT_79d: Attribute / MF / DF.strukturiert 557
Tabelle 394: CosT_f1d: Attribute / MF / DF.strukturiert / EF.strukturiert 558
Tabelle 395: CosT_64f: Attribute / MF / DF.transparent 559
Tabelle 396: CosT_400: Attribute / MF / DF.transparent / EF.transparent 559
Tabelle 397: CosT_f68, Kurvenparameter ansix9p256r1 560
Tabelle 398: CosT_88a: Kurvenparameter ansix9p384r1 561
Tabelle 399: CosT_de0: Kurvenparameter brainpoolP256r1 562
Tabelle 400: CosT_62f: Kurvenparameter brainpoolP384r1 562
Tabelle 401: CosT_198: Kurvenparameter brainpoolP512r1 563
Tabelle 402: CosT_039: Beispiel der zeitlichen Abfolge asynchroner Administration 588

21.5 Referenzierte Dokumente

21.5.1 Dokumente der gematik

Die nachfolgende Tabelle enthält die Bezeichnung der in dem vorliegenden Dokument referenzierten Dokumente der gematik zur Telematikinfrastruktur. Version und Stand der referenzierten Dokumente sind in der nachfolgenden Tabelle nicht aufgeführt. Deren zu diesem Dokument passende jeweils gültige Versionen sind in den von der gematik veröffentlichten Produkttypsteckbriefen enthalten, in denen die vorliegende Version aufgeführt wird.

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
~~[gemSpec_eGK_P1]~~	~~gematik: Teil 1 – Spezifikation der elektrischen Schnittstelle (Vorgängerversion dieses Dokumentes für Karten der Generation 1)~~
[gemSpec_Krypt]	gematik: Übergreifende Spezifikation - Verwendung kryptographischer Algorithmen in der Telematikinfrastruktur
[gemSpec_eGK_ObjSys]	gematik: Spezifikation der elektronischen Gesundheitskarte eGK-Objektsystem
[gemSpec_HBA_ObjSys]	gematik: Spezifikation des elektronischen Heilberufsausweises HBA-Objektsystem
[gemSpec_OID]	gematik: Festlegung von OIDs
[gemSpec_PKI]	gematik: Übergreifende Spezifikation, Spezifikation PKI
[gemSpec_SMC-B_ObjSys]	gematik: Spezifikation der Secure Module Card SMC-B Objektsystem

21.5.2 Weitere Dokumente

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
~~[ANSI X3.92]~~	~~American National Standard X3.92 – 1981, Data Encryption Algorithm~~
[ANSI X9.62]	Public Key Cryptography for the Financial Services Industry, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 2005

[ANSI X9.63]	Public Key Cryptography for the Financial Services Industry, *Key Agreement and Key Transport Using Elliptic Curve Cryptography*, 2001
[BinPrefix]	Prefix for binary multiples IEC INTERNATIONAL STANDARD 60027-2, Third edition, 2005-08, Letter symbols to be used in electrical technology – Part 2: Telecommunications and electronics, clause 3.8.3 http://physics.nist.gov/cuu/Units/binary.html http://de.wikipedia.org/wiki/Bin%C3%A4rpr%C3%A4fix http://en.wikipedia.org/wiki/Binary_prefix
~~[BKryA]~~	~~Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen), Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, veröffentlicht am 18. Januar 2012 im Bundesanzeiger, Nr. 10, S. 243 (auch online verfügbar: http://www.bundesnetzagentur.de)~~
[CMAC]	NIST Special Publication 800-38B, Recommendation for Block, Cipher Modes of Operation: The CMAC Mode for Authentication, Morris Dworkin, May 2005, http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
[PP COS G2]	Bundesamt für Sicherheit in der Informationstechnik, Common Criteria Protection Profile Card Operating System Generation 2, BSI-CC-PP-0082-V4, Version 2.1, 10. July 2019
[EMV®_Book-1]	EMV Book 1, Application Independent ICC to Terminal Interface Requirements, version 4.3, November 2011
[ETSI TS 102 222]	ETSI TS 102 222 V7.0.0 (2006-08) Integrated Circuit Cards (ICC); Administrative commands for telecommunications applications (Release 7)
[FIPS 180-4]	Federal Information Processing Standards Publication 180-4 Secure Hash Standard (SHS), March 2012 http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf
[FIPS 186–4]	FIPS PUB 186-4, FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, July 2013 DIGITAL SIGNATURE STANDARD (DSS) http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
[FIPS 197]	Federal Information Processing Standards Publication 197, (FIPS–197), November 26, 2001, Announcing the ADVANCED ENCRYPTION STANDARD (AES) http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
~~[HBA_P1]~~	~~Spezifikation des elektronischen Heilberufsausweises Teil I: Kommandos, Algorithmen und Funktionen der Betriebssystemplattform,~~

	~~Version 2.3.2, 05.08.2009~~ ~~(Vorgängerversion dieses Dokumentes für Karten der Generation 1)~~
[ISO/IEC 7816-3]	ISO/IEC 7816-3: 2006 (2nd edition), Identification cards — Integrated circuit cards with contacts — Part 3: Electrical interface and transmission protocols
[ISO/IEC 7816-4]	ISO/IEC 7816-4: 2013 (3rd edition) Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange
[ISO/IEC 7816-5]	ISO/IEC 7816-5: 2004 (2nd edition) Identification cards — Integrated circuit cards — Part 5: Registration of application providers
[ISO/IEC 7816-8]	ISO/IEC 7816-8: 2019 (4th edition) Identification cards — Integrated circuit cards — Part 8: Commands and mechanisms for security operations
[ISO/IEC 7816-9]	ISO/IEC 7816-9: 2017 (3rd edition) Identification cards — Integrated circuit cards — Part 9: Commands for card management
[ISO/IEC 7816-12]	ISO/IEC 7816-12: 2005-10-04 (1st edition) Identification cards — Integrated circuit cards — Part 12: Cards with contacts — USB electrical interface and operating procedures
[ISO/IEC 7816-13]	ISO/IEC 7816-13: 2007 (1st edition) Identification cards — Integrated circuit cards — Part 13: Commands for application management in a multi-application environment
[ISO/IEC 9796-2]	ISO/IEC 9796-2: 2010-12-15 (3rd edition) Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms
[ISO/IEC 14443-1]	ISO/IEC 14443-1: 2016-03 (3rd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 1: Physical characteristics
[ISO/IEC 14443-2]	ISO/IEC 14443-2: 2016-07 (3rd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 2: Radio frequency power and signal interface
[ISO/IEC 14443-3]	ISO/IEC 14443-3: 2016-06 (3rd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 3: Initialization and anticollision

[ISO/IEC 14443-4]	ISO/IEC 14443-4: 2016-06 (3rd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 4: Transmission protocol
[ITU-T X.690]	ITU-T X.690, 2008-11, entspricht ISO/IEC 8825-1 Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
[NIST sp800-38a]	Section 6.5 of NIST Special Publication 800-38A, *Recommendation for Block, Cipher Modes of Operation, Methods and Techniques*, Morris Dworkin, December 2001 Edition, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
[PKCS#1]	PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf
[RFC2119]	Key words for use in RFCs to Indicate Requirement Levels, RFC 2119, March 1997, http://tools.ietf.org/html/rfc2119
[RFC5639]	Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, RFC 5639, March 2010, http://tools.ietf.org/html/rfc5639
[BSI-TR-03110-2]	Technical Guideline TR-03110-2, Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 2 – Protocols for electronic IDentification, Authentication and trust Services (eIDAS)~~Extended Access Control Version 2 (EACv2), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI)~~, Version 2.21, 2016-12-21
[BSI-TR-03110-3]	Technical Guideline TR-03110-3, Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 3 – Common Specifications, Version 2.21, 2016-12-21
[BSI-TR-03111]	Technical Guideline TR-03111, Elliptic Curve Cryptography, Version 2.10, 2018-06-01
[BSI-TR-03116-1]	Technische Richtlinie BSI TR-03116-1, Kryptographische Vorgaben für Projekte der Bundesregierung, Teil 1: Telematikinfrastruktur, Version 3.20, Datum 21.09.2018, Status: Veröffentlichung, Fassung September 2018

22 Asynchrone, gesicherte APDU-Sequenzen (informativ)

22.1 Einleitung

Gemäß einer Anforderung an das COS der Generation 2 ist es einem Kartenmanagement (CMS) möglich nicht nur synchron, sondern auch asynchron mit Karten der Generation 2 zu kommunizieren. Dieses Papier beleuchtet die diesbezügliche Anforderungslage und zeigt eine technische Umsetzungsmöglichkeit.

22.2 CMS-Kommunikationsmuster

Hier werden das synchrone und das asynchrone Kommunikationsmuster zwischen einem Kartenmanagement und einer Smartcard aus technischer Sicht betrachtet.

22.2.1 Synchrone Kommunikation zwischen CMS und Smartcard

Das wesentliche Merkmal der synchronen Kommunikation ist, dass der Sender einer Nachricht auf eine Antwort wartet, bevor die nächste Nachricht versendet wird.



Abbildung 13: CosA_88c: Synchrone Kommunikation zwischen CMS und Smartcard

Aus technischer wie aus sicherheitstechnischer Sicht sind hier im Rahmen der synchronen Kommunikation lediglich die beiden Komponenten CMS und Smartcard relevant. Dass die Kommunikationsstrecke über Kartenleser, Router oder Internet führt, ist hier unerheblich. Für Karten der Generation 1 steht am Anfang einer CMS-Prozedur eine gegenseitige Authentisierung mit Sessionkey-Aushandlung. Da an der Sessionkey-Berechnung beide Kommunikationspartner beteiligt sind, ist es erst nach Abschluss der gegenseitigen Authentisierung möglich das vom CMS zu sendende Szenario (Abfolge von Kommando-APDU) mit den ausgehandelten Sessionkeys zu schützen.

22.2.2 Asynchrone Kommunikation zwischen CMS und Smartcard

Das wesentliche Merkmal der asynchronen Kommunikation ist, dass der Sender einer Nachricht nicht auf eine Antwort wartet. Ein Blockieren des Senders bis zum Empfang der zugehörigen Antwort findet nicht statt.

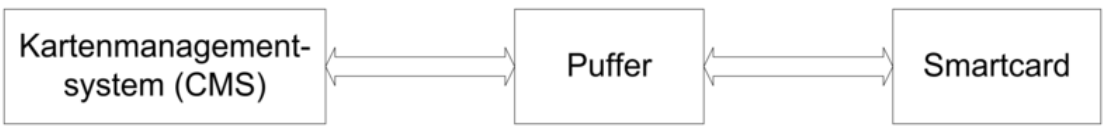


Abbildung 14: CosA_e47: Asynchrone Kommunikation zwischen CMS und Smartcard

Aus technischer, wie aus sicherheitstechnischer Sicht sind hier im Rahmen der asynchronen CMS-Kommunikation drei Komponenten beteiligt:

1. Das CMS erstellt Szenarien (Abfolgen von Kommando-APDU) und bietet diese über eine Schnittstelle an (beispielsweise zum Download).
2. Ein Puffer besorgt sich zu einem beliebigen späteren Zeitpunkt T1 beliebige Szenarien. Zu einem beliebigen Zeitpunkt T2 (wobei im Allgemeinen T1 ungleich T2 ist) werden ein oder mehrere Szenarien vom Puffer an eine Smartcard geschickt.
3. Die Smartcard verarbeitet die Szenarien und erstellt dabei typischerweise "Quittungen", die eine erfolgreiche Verarbeitung anzeigen. Diese "Quittungen" können vom Puffer zu einem beliebigen späteren Zeitpunkt an das CMS übertragen werden.

Für die Komponente Puffer gibt es diverse Realisierungsmöglichkeiten. Im einen Extrem befindet sich der Puffer möglichst nah an der Smartcard, beispielsweise versendet das CMS per Email Szenarien etwa an einen Karteninhaber. Dann wäre "Puffer" aus Sicht des Karteninhabers ein lokales Programm. Im anderen Extrem befindet sich der Puffer möglichst nah am CMS, beispielsweise wäre der Puffer lediglich der per Internet erreichbare Online-Teil des CMS und der übrige CMS Teil (mit sensiblen, personenbezogenen Daten und Schlüsseln) wäre nicht per Internet erreichbar.

22.3 Anforderungen an die asynchrone Kommunikation

Aus sicherheitstechnischer Sicht werden an die asynchrone Kommunikation folgende Anforderungen gestellt:

1. Asynchrone Kommunikation: Es MUSS dem CMS möglich sein komplette Szenarien so vorzuberechnen, dass das CMS beim Einspielen der Szenarien in eine Smartcard nicht beteiligt ist.
2. Quittungen: Es MUSS möglich sein, dass die Smartcard für einzelne oder alle Kommando-APDU eines Szenarios Quittungen so erzeugt, dass sich das CMS zweifelsfrei von einer erfolgreichen Kommandobearbeitung überzeugen kann.
3. Authentizität: Das CMS MUSS sich zweifelsfrei als Quelle eines Szenarios und jeder einzelnen Kommando-APDU innerhalb eines Szenarios identifizieren lassen.
4. Integrität eines Szenarios: Jede Veränderung eines Szenarios und jeder einzelnen Kommando-APDU MUSS sicher erkennbar sein. Typische Veränderungen sind:
   a. Verändern einer einzelnen Kommando-APDU
   b. Hinzufügen beliebiger Kommando-APDU (auch Duplizieren von Kommando-APDU)
   c. Weglassen beliebiger Kommando-APDU
   d. Verändern der Reihenfolge von Kommando-APDU
5. Vertraulichkeit: Es MUSS möglich sein, die Vertraulichkeit von Informationen zu gewährleisten, die in einem Szenario transportiert werden.

6. Reihenfolge von Szenarien: Es MUSS möglich sein, die Reihenfolge von Szenarien vorzugeben, beispielsweise: Szenario_B nur, wenn auch Szenario_A durchgeführt wurde. Relevant wird diese Anforderung etwa dann, wenn zunächst eine Anwendung geladen wird und später die Anwendungsdaten aktualisiert werden. Eine Aktualisierung von Anwendungsdaten ist nur dann sinnvoll, wenn die Anwendung geladen wurde.
7. Resistenz gegen Replay Attacken: Es ist auszuschließen, dass ein Szenario ein zweites Mal erfolgreich durchgeführt wird.
8. Überspringen von Szenarien: Es MUSS dem CMS möglich sein Szenarien als obsolet zu kennzeichnen. Relevant wird diese Anforderung etwa dann, wenn sich Anwendungsdaten regelmäßig oder rasch ändern. Anstatt dann zunächst veraltete Szenarien an die Smartcard zu schicken, ist es dann möglich gleich das aktuelle Szenario auszuführen. Konkret:
   a. Regelmäßige Änderung von Anwendungsdaten: CV-Zertifikate mit Gültigkeitszeiträumen von wenigen Tagen sind regelmäßig zu aktualisieren. Nach einem mehrwöchigen Urlaub ist es sinnvoller gleich das aktuelle CV-Zertifikat zu laden und alle während des Urlaubs angefallenen Zwischenschritte der CVC-Aktualisierung zu überspringen.
   b. Rasche Änderung von Anwendungsdaten: In gewissen Situationen ändern sich die Versichertendaten möglicherweise mehrmals kurz hintereinander (etwa ändere sich durch eine Heirat zunächst der Familienstatus und durch den Bezug einer gemeinsamen Wohnung dann die Adresse, oder nach Abschluss einer Ausbildung / Studium ändert sich zunächst der Versichertenstatus und dann durch einen Wohnortwechsel im Zusammenhang mit einer Arbeitsaufnahme die Adresse, etc.). Dann erscheint es sinnvoller veraltete Szenarien zu überspringen.

22.4 Lösungskonzept

Zur Umsetzung der in 22.3 vorgeschlagenen Anforderungen wird folgendes Konzept vorgeschlagen:

1. Kryptographisches Material: Sowohl CMS, als auch jede vom CMS betreute Smartcard verfügen über kryptographisches Material. Das CMS weist damit seine Authentizität nach (Anforderung Authentizität). Zudem wird es zum sicheren Transport von Sessionkeys hin zur Smartcard verwendet (Anforderung asynchrone Kommunikation). Sessionkeys und SendSequenceCounter schützen die Kommando-APDU (Anforderungen Authentizität, Integrität eines Szenarios, Vertraulichkeit) eines Szenarios sowie die Antwort-APDU einer Smartcard (Anforderung Quittungen).
2. NumberScenario: Jedem Szenario wird vom CMS eine natürliche Zahl zugeordnet. Die Zuordnung zwischen Szenario und NumberScenario MUSS bijektiv (d.h. umkehrbar eindeutig) sein.
3. NumberPrecondition: Jedem Szenario wird vom CMS eine weitere natürliche Zahl oder die null zugeordnet. Diese gibt an, welches Szenario als Vorbedingung in eine Smartcard geladen worden sein muss, damit dieses Szenario akzeptiert wird

(siehe Beispiel) (Anforderung Reihenfolge von Szenarien, Überspringen von Szenarien).

4. Die Smartcard speichert NumberScenario eines Szenarios. Szenarien MÜSSEN so an die Smartcard geschickt werden, dass ihr NumberScenario streng monoton steigt (Anforderungen Resistent gegen Replay Attacken). Zudem MUSS NumberPrecondition eines Szenarios kleiner oder gleich dem in der Smartcard gespeicherten Wert von NumberScenario sein (Anforderung Reihenfolge von Szenarien, Überspringen von Szenarien).

Im folgenden Beispiel gelte folgende Notation: $S_{NumberPrecondition}$(*NumberScenario*). Dem Szenario S wird damit die natürliche Zahl *NumberScenario* und die Vorbedingung *NumberPrecondition* zugeordnet. Die folgende Tabelle stellt eine Chronologie dar, wobei die Zeit mit der Zeilennummer zunehme. Die Spalte CMS enthält Szenarien, die vom CMS zum betrachteten Zeitpunkt bereitgestellt werden. Die Spalte Puffer enthält die dort gespeicherten Szenarien. Szenarien sind grün hinterlegt, wenn sie von einer Smartcard akzeptiert werden und rot hinterlegt, wenn sie von einer Smartcard wegen unpassender *NumberPrecondition* oder unpassender *NumberScenario* nicht akzeptiert werden. Gelb hinterlegte Szenarien werden im betrachteten Zeitpunkt an die Smartcard geschickt.

Tabelle 402: CosT_039: Beispiel der zeitlichen Abfolge asynchroner Administration

CMS	Puffer	Smartcard	Bemerkung
-	-	*NumberScenario* = 0	Zu einem bestimmten Zeitpunkt sei eine Smartcard im Feld und der Wert von *NumberScenario* sei 0.
$S_0(1)$	$S_0(1)$	-	Das CMS stellt ein Szenario zur Verfügung, welches vom Puffer heruntergeladen werde. Die Smartcard ist inaktiv.
$S_0(2)$	$S_0(1)$ $S_0(2)$	-	Das CMS ersetzt $S_0(1)$ durch $S_0(2)$. Durch das Ersetzen und da *NumberPrecondition* unverändert auf 0 steht wird $S_0(1)$ als obsolet gekennzeichnet. Der Puffer lädt zusätzlich $S_0(2)$ herunter. Die Smartcard ist inaktiv.
$S_0(2)$ $S_2(3)$	$S_0(1)$ $S_0(2)$ $S_2(3)$	-	Das CMS stellt $S_2(3)$ zusätzlich zu $S_0(2)$ zur Verfügung und der Puffer lädt auch dieses herunter. *NumberPrecondition* in $S_2(3)$ ist so gesetzt, dass erst $S_0(2)$ eingespielt werden muss, bevor $S_2(3)$ einspielbar ist. Die Smartcard ist weiter inaktiv.
$S_0(2)$ $S_2(4)$	$S_0(1)$ $S_0(2)$ $S_2(3)$ $S_2(4)$	-	Das CMS ersetzt $S_2(3)$ durch $S_2(4)$. Durch das Ersetzen und da *NumberPrecondition* unverändert auf 2 steht, wird $S_2(3)$ als obsolet gekennzeichnet. Der Puffer lädt zusätzlich $S_2(4)$ herunter. Die Smartcard ist inaktiv.
$S_0(2)$ $S_2(4)$	$S_0(1)$ $S_0(2)$ $S_2(3)$ $S_2(4)$	$S_0(1)$ zur Karte *NumberScenario* = 1	Der Puffer schicke $S_0(1)$ an die Smartcard. Dies ist suboptimal (aber unschädlich), da $S_0(1)$ durch $S_0(2)$ obsolet wurde.
$S_0(2)$ $S_2(4)$	$S_0(1)$ $S_0(2)$ $S_2(3)$ $S_2(4)$	$S_0(2)$ zur Karte *NumberScenario* = 2	Der Puffer schicke das einzig akzeptable Szenario $S_0(2)$ an die Smartcard. $S_0(1)$ ist nicht akzeptabel, weil dessen *NumberScenario* nicht größer ist als *NumberScenario* der Smartcard. $S_2(3)$ ist nicht akzeptabel, da dessen *NumberPrecondition* nicht kleiner ist als *NumberScenario* der Smartcard.
$S_0(2)$ $S_2(4)$	$S_0(1)$ $S_0(2)$ $S_2(3)$ $S_2(4)$	$S_2(4)$ zur Karte *NumberScenario* = 4	Diesmal arbeite der Puffer optimal und schicke das aktuellste akzeptable Szenario an die Smartcard. Zudem schicke der Puffer nach Abschluss des Szenarios die zu $S_2(4)$ gehörende Quittung an das CMS.
-	$S_0(1)$ $S_0(2)$ $S_2(3)$ $S_2(4)$	*NumberScenario* = 4	Nach Empfang der Quittung können die Szenarien aus dem Downloadbereich des CMS entfernt werden. Keines der bisher vom CMS bereitgestellten Szenarien ist für die Smartcard akzeptabel.

22.5 Kryptographie

Zur Erfüllung der in 22.3 genannten kryptographischen Anforderungen werden symmetrische und asymmetrische Authentisierungsschlüssel verwendet.

22.5.1 CMS Aktivitäten

Auf Seiten des CMS ist der Ablauf wie folgt:

5. Erstellen des gesicherten Szenarios, hierfür sind keine kartenindividuellen Informationen erforderlich:
   a. Zu Zwecken des Kartenmanagements wird eine Sequenz von ungesicherten Kommando-APDU erstellt.
   b. Das CMS erzeugt einen Seed zur Ableitung von AES-Sessionkeys und entsprechenden Werten für die SendSequenceCounter für die CMAC-Berechnung.
   c. Die ungesicherten Kommando-APDU werden mittels der Sessionkeys und SendSequenceCounter konform zu 13.1.2 gesichert.
6. Kartenindividuelles Szenario:
   a. Das CMS besorgt sich (aus einer Datenbank oder ähnlichem) kartenspezifische Werte für ICCSN, NumberPrecondition, NumberScenario und das Schlüsselmaterial:
      i. symmetrische Administration: SK.CMS_ENC, SK.CMS_MAC.
      ii. asymmetrische Administration: PuK.SmartCard.AUT, PrK.CMS, PuK.CMS.
   b. Der Seed aus Schritt 16)b) wird verschlüsselt:
      i. symmetrische Administration:
      cipherText = AES_CBC_ENC(SK.CMS_ENC, ICV='00…00', Seed).
      ii. asymmetrische Administration:
         A. (POA, C, T) = ELC_ENC(Seed, PuK.SmartCard.AUT, PuK.CMS.dP)
         B. Setze oidDO = '06 – L06 – affectedObject.domainParameter.OID'.
         C. Setze keyDO = '7F49 – L7F49 – (86 – L86 – POA)'.
         D. Setze cipherDO = '86 – L86 – (02 || C)'.
         E. Setze macDO = '8E – L8E – T '.
         F. Setze cipherText = 'A6 – LA6 – (oidDO || keyDO || cipherDO || macDO) '.
   c. Es wird eine Nachricht M wie folgt zusammengestellt:
      M = NumberPrecondition || NumberScenario || cipherText.
   d. Für die Nachricht M werden Authentisierungsdaten berechnet:
      i. symmetrische Administration: Es wird ein CMAC für M berechnet:
      CMAC = CalculateCMAC_IsoPadding(SK.CMS_MAC, M).
      ii. asymmetrische Administration: Es wird eine Signatur für M berechnet:
         A. Wenn PrK.CMS.dP = brainpool256 => hash = SHA_256(M).
         B. Wenn PrK.CMS.dP = brainpool384 => hash = SHA_384(M).
         C. Wenn PrK.CMS.dP = brainpool512 => hash = SHA_512(M).

D. (R, S) = ELC_SIG(PrK.CMS, hash)

E. SIG = R || S.

e. Eine Authentisierungssequenz mit M und CMAC bzw. SIG wird dem Szenario aus Schritt 16)c) vorangestellt.

22.5.2 Puffer-Aktivitäten

Auf Seiten des Puffers ist der Ablauf im Gutfall wie folgt:

7. Der Puffer ermittelt von einer Smartcard die Werte ICCSN und NumberScenario.
8. Der Puffer wählt ein passendes Szenario aus.
9. Der Puffer schickt das ausgewählte Szenario an die Smartcard.
10. Der Puffer schickt die gesicherten Antwort-APDU zusammen mit ICCSN und NumberScenario als Quittung an das CMS.

22.5.3 Smartcard-Aktivitäten

Auf Seiten der Smartcard ist der Ablauf wie folgt:

11. Schlüsselauswahl: Zu Beginn des Szenarios werden ein passendes Schlüsselobjekt und ein passender Algorithmus ausgewählt (etwa durch ein MANAGE SECURITY ENVIRONMENT Set-Kommando).
12. Authentisierung: Die Authentisierungssequenz aus Schritt 17)e) folgt im Szenario als nächstes. Dabei werden M und CMAC bzw. SIG übertragen und in der Smartcard wie folgt verarbeitet:
    a. Der CMAC bzw. SIG wird überprüft. Ein unpassender CMAC oder eine fehlerhafte Signatur führt zum Kommandoabbruch.
    b. Der Nachricht M werden NumberPrecondition, NumberScenario und cipherText entnommen.
        i. Wenn NumberPrecondition in M größer als SK.CMS→*numberScenario* bzw. PrK.SmartCard.AUT→*numberScenario* ist, dann führt dies zum Kommandoabbruch.
        ii. Wenn NumberScenario in M kleiner oder gleich SK.CMS→*numberScenario* bzw. PrK.SmartCard.AUT→*numberScenario* ist, dann führt dies zum Kommandoabbruch.
    c. SK.CMS→*numberScenario* bzw. PrK.SmartCard.AUT→*numberScenario* wird mit Transaktionsschutz auf den Wert von NumberScenario aus M gesetzt.
    d. cipherText wird mittels SK.CMS_ENC bzw. PrK.SmartCard.AUT entschlüsselt und Seed wird zum Berechnen von Sessionkeymaterial inklusive SendSequenceCountern verwendet.
13. Kartenmanagementkommandos: Die übrigen, Secure Messaging gesicherten Kommando-APDU werden an die Smartcard geschickt. Secure-Messaging-Fehler führen (wie üblich) zum Kommandoabbruch.

22.6 Auswirkungen auf die Kommandoschnittstelle

Das hier vorgestellte Konzept der asynchronen Kommunikation zwischen CMS und Smartcard wirkt sich an zwei Stellen auf die Smartcard aus: Zum einen muss es dem Puffer möglich sein NumberScenario aus der Smartcard auszulesen, damit das passende Szenario ausgewählt werden kann. Zum anderen gilt es den Sessionkeykontext in der Smartcard zu etablieren.

1. RESET der Smartcard
2. MANAGE SECURITY ENVIRONMENT Set:
   a. P1P2='81A4',
   b. *keyReference* = SK.CMS oder PrK.SmartCard.AUT,
   c. *algID*=asynchroneCMS
3. GENERAL AUTHENTICATE Datenfeld so, dass NumberScenario von der Smartcard angefordert wird.
4. GENERAL AUTHENTICATE Datenfeld mit M und CMAC (oder M und Signatur), ein passender CMAC (bzw. eine gültige Signatur) setzt einen Sicherheitszustand. Mit den Informationen aus M werden Sessionkeys und SendSequenceCounter eingerichtet.
5. Der Rest des Szenarios wird mit Secure Messaging geschützt an die Smartcard geschickt.

23 Speichern öffentlicher Schlüssel (informativ)

Dieser Anhang beschreibt, wie öffentliche Schlüsselobjekte (siehe 8.6.4) bezüglich ihrer Einordnung in ein Objektsystem in diesem Dokument betrachtet werden. Einerseits werden öffentliche Schlüsselobjekte genauso wie andere Objekte auch einem Ordner zugeordnet und damit in die Objektsystemhierarchie eingeordnet wie die übrigen Objekte. Zudem werden sie aus funktionaler Sicht analog zu anderen Schlüsselobjekten mittels MANAGE SECURITY ENVIRONMENT selektiert und im Rahmen einer Kommandobearbeitung gesucht und angesprochen. Andererseits werden bestimmte öffentliche Schlüsselobjekte mittels CV-Zertifikaten dynamisch zur Laufzeit in eine Smartcard importiert und es wäre unpassend, wenn dadurch der gesamte freie Speicher verbraucht würde. Deshalb ist das Importieren mittels CV-Zertifikaten aus Sicht des Speichermanagements anders zu betrachten, als etwa das Einbringen neuer Objekte mittels LOAD APPLICATION.

Dieser Anhang und die normativen Teile dieses Dokumentes beschreiben öffentliche Schlüsselobjekte aus diversen Perspektiven mit dem Ziel die Funktionalität an der äußeren Kartenschnittstelle zu verdeutlichen und normativ festzuschreiben. Hier, wie auch im Rest des Dokumentes, ist es nicht das Ziel eine bestimmte Implementierung zu fordern oder zu präferieren.

23.1 Definitionen

allPublicKeyList: Ist die Vereinigungsmenge von *applicationPublicKeyList* und Cache.

applicationPublicKeyList, (N019.900)d: Liste zur persistenten Speicherung von öffentlichen Schlüsselobjekten der Anwendungen. Im Rahmen dieser Betrachtung sind das öffentliche Schlüsselobjekte, die im Rahmen einer Initialisierung oder als Teil einer Ordnerstruktur mittels LOAD APPLICATION in die Smartcard geladen werden.

Cache: Bereich zur temporären, möglicherweise persistenten Speicherung von Schlüsselobjekten, die mittels CV-Zertifikaten importiert wurden. Vereinigungsmenge von *persistentCache* und *volatileCache*.

CA-Schlüssel: Öffentlicher Signaturprüfschlüssel einer CVC-Sub-CA, welcher mittels CV-Zertifikat in eine Karte importiert wird.

EE-Schlüssel: Öffentlicher Authentisierungsschlüssel eines Endnutzers (**E**nd-**E**ntity), welcher mittels CV-Zertifikat in eine Karte importiert wird.

persistentCache, (N019.900)e: Teil des Cache der persistent gespeichert ist.

persistentPublicKeyList: Vereinigungsmenge von *applicationPublicKeyList* und *persistentCache*.

Sicherheitsanker: Öffentliches Signaturprüfobjekt, welches den öffentlichen Schlüssel einer CVC-Root-CA enthält. Typischerweise werden Sicherheitsanker im Rahmen einer Initialisierung oder Personalisierung in eine Smartcard geladen. Zusätzlich ist es möglich Sicherheitsanker mittels Link-Zertifikaten in die Karte zu importieren.

volatileCache, (N019.900)g: Teil des Cache der volatil gespeichert ist.

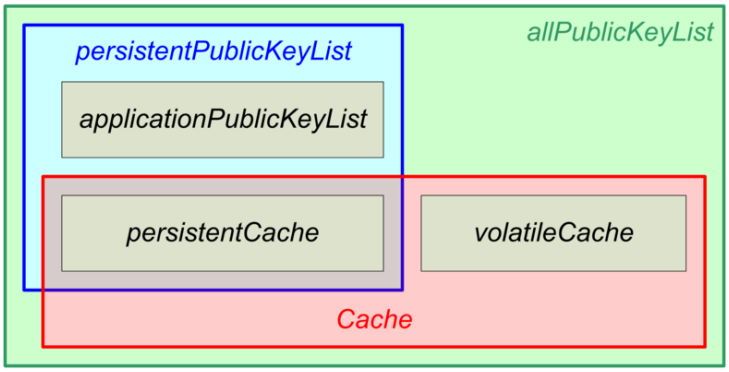


Abbildung 15: CosA_85c: Zusammenhang Schlüsselspeicher

23.2 Perspektiven

Aus der Perspektive der Kommandobearbeitung wird in diversen Kommandos zunächst nach einem Objekt gesucht, auf welches sich das Kommando bezieht. Bezogen auf öffentliche Schlüsselobjekte fasst dieses Dokument alle öffentlichen Schlüsselobjekte in einer Liste *allPublicKeyList* zusammen.

Aus der Perspektive des Nachladens von Ordnern mittels LOAD APPLICATION werden öffentliche Schlüsselobjekte, die im Ordner enthalten sind, in der Sichtweise dieses Dokumentes *applicationPublicKeyList* hinzugefügt. Es wird davon ausgegangen, dass *applicationPublicKeyList* nur eine begrenzte Kapazität hat. Deshalb ist es denkbar, dass ein LOAD APPLICATION aus Platzmangel scheitert, wenn die Menge der nachgeladenen Schlüsselobjekte die Kapazität von *applicationPublicKeyList* übersteigt.

Aus der Perspektive des Importes mittels CV-Zertifikaten werden importiere Schlüssel dem Cache hinzugefügt. Es wird davon ausgegangen, dass der Cache nur eine begrenzte Kapazität hat. Wenn die Menge an importierten Schlüsseln die Kapazität des Cache übersteigt, dann werden "unwichtige" Einträge aus dem Cache entfernt um Platz für den importierten Schlüssel zu schaffen.