

Elektronische Gesundheitskarte und Telematikinfrastruktur

Spezifikation des Card Operating System (COS)

Elektrische Schnittstelle

Version: 3.11.0
Revision: \main\rel_opb1\rel_ors2\1
Stand: 14.05.2018
Status: freigegeben
Klassifizierung: öffentlich
Referenzierung: [gemSpec_COS]

Dokumentinformationen

Es handelt sich hier um eine veränderte Fassung des Dokumentes für Karten der Generation 2.1.

Die wichtigsten Änderungen gegenüber der Betriebssystemspezifikation der Generation 2 sind:

- 1) Der DES Algorithmus wurde aus dem normativen Umfang in ein optionales Funktionspaket verschoben, siehe (N000.030), entsprechende Performanzvorgaben wurden entfernt.
- 2) CV-Zertifikate basierend auf dem RSA Algorithmus wurden in ein optionales Funktionspaket verschoben, siehe (N000.032) entsprechende Performanzvorgaben wurden entfernt.
- 3) 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.
- 4) Anforderungen zur Kürzung von *keyReferenceList* im Kommando LIST PUBLIC KEY präzisiert, siehe 14.9.7.3.
- 5) Die RSA Schlüsselgenerierung durch das COS wurde aus dem normativen Umfang in ein optionales Funktionspaket verschoben, siehe (N000.034).

Die wichtigsten Änderungen gegenüber den Betriebssystemspezifikationen der Generation 1 sind:

- 6) Alle Betriebssystemspezifikationen der Generation 1 wurden in diesem Dokument zusammengefasst.
- 7) Alle SRQs zu den Betriebssystemspezifikationen wurden integriert.
- 8) Die kontaktlose Datenübertragung und die Datenübertragung gemäß [ISO/IEC 7816-12] wurden aufgenommen.
- 9) Die Kryptographie gemäß [FIPS 197] (AES) und [BSI-TR-03111] (elliptische Kurven) wurden dem normativen Teil hinzugefügt.
- 10) Für RSA-Schlüssel ist die Modulslänge 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 Modulslänge 3072 bit	gematik
3.1.0	30.01.13		Anpassung des normativen Umfangs, Fehlerkorrekturen	gematik
	16.05.13		Fehlerkorrekturen	PL P71
	20.09.13		Präzisierung PACE Einarbeitung von Kommentaren	PL P71
3.2.0	22.10.13		Einarbeitung von Kommentaren	gematik
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
	09.03.18		RSA Schlüsselgenerierung aus normativem Umfang entfernt	TEC/TN
3.11.0	14.05.18		freigegeben	gematik

Inhaltsverzeichnis

Dokumentinformationen	2
Inhaltsverzeichnis	4
1 Einordnung des Dokuments	21
1.1 Zielsetzung	21
1.2 Zielgruppe	22
1.3 Geltungsbereich	22
1.4 Abgrenzung des Dokuments	23
1.5 Methodik	23
1.5.1 Nomenklatur der Präfixe	23
1.5.2 Nomenklatur Verschiedenes	24
1.5.3 Normative und informative Abschnitte	24
1.5.4 Komponentenspezifische Anforderungen	25
1.5.5 Verwendung von Schlüsselworten	26
2 Optionen (normativ)	27
3 Systemüberblick (informativ)	29
4 Lebenszyklus von Karte und Applikation (informativ)	30
5 Datentypen und Datenkonvertierung (normativ)	31
5.1 BitLength Anzahl Bit in einem Bitstring	31
5.2 OctetLength Anzahl Oktett in einem Oktettstring	32
5.3 I2OS Integer nach Oktettstring	32
5.4 OS2I Oktettstring nach Integer	33
5.5 OS2P Oktettstring nach Punkt (uncompressed encoding)	33
5.6 P2OS Endlicher Punkt nach Oktettstring	34
5.7 Extrahiere führende Elemente	35
5.7.1 Extrahiere führende Bits	35
5.7.2 Extrahiere führende Oktette	35
5.8 PaddingIso	35
5.9 TruncateIso	36
5.10 MGF Mask Generation Function	37
5.11 RAND Zufälliger Oktettstring	38
5.12 ceiling Aufrunden einer reellen Zahl	38

5.13	floor Abrunden einer reellen Zahl.....	38
6	Kryptographische Algorithmen (normativ).....	39
6.1	Hash-Algorithmen.....	39
6.1.1	SHA-1.....	39
6.1.2	SHA-256.....	39
6.1.3	SHA-384.....	40
6.1.4	SHA-512.....	40
6.2	Schlüsselvereinbarung	40
6.2.1	Verhalten für 3DES-Schlüssel, Option_DES.....	40
6.2.2	Vereinbarung von AES-128-Schlüsseln.....	41
6.2.3	Vereinbarung von AES-192-Schlüsseln.....	41
6.2.4	Vereinbarung von AES-256-Schlüsseln.....	42
6.2.5	Schlüsselableitung aus einer Card Access Number.....	42
6.3	Symmetrischer Basialgorithmus für Vertraulichkeit.....	43
6.3.1	Symmetrische Verschlüsselung eines Datenblocks, Option_DES	43
6.3.1.1	Verschlüsselung mittels DES, Option_DES.....	43
6.3.1.2	Verschlüsselung mittels 3DES, Option_DES	44
6.3.2	Symmetrische Entschlüsselung eines Datenblocks, Option_DES.....	44
6.3.2.1	Entschlüsselung mittels DES, Option_DES.....	44
6.3.2.2	Entschlüsselung mittels 3DES, Option_DES.....	44
6.3.3	Symmetrische Verschlüsselung eines Datenblocks, AES.....	45
6.3.4	Symmetrische Entschlüsselung eines Datenblocks, AES	45
6.4	Asymmetrischer Basialgorithmus RSA	46
6.5	Asymmetrischer Basialgorithmus elliptische Kurven.....	47
6.6	Datenauthentisierung	47
6.6.1	MAC-Generierung	47
6.6.1.1	Generierung Retail-MAC, Option_DES	47
6.6.1.2	Generierung CMAC ohne internes Padding vor der CMAC-Berechnung 48	
6.6.1.3	Generierung CMAC mit internem ISO-Padding vor der CMAC- Berechnung.....	49
6.6.2	MAC-Prüfung.....	49
6.6.2.1	Prüfung Retail-MAC, Option_DES	49
6.6.2.2	Prüfung CMAC ohne internes Padding vor der CMAC-Berechnung.....	49
6.6.2.3	Prüfung CMAC mit internem Padding vor der CMAC Berechnung.....	50
6.6.3	Signaturberechnung	50
6.6.3.1	Signaturberechnung mittels RSA	50
6.6.3.1.1	RSA, ISO9796–2, DS1, SIGN, Option_RSA_CVC.....	50
6.6.3.1.2	RSA, SSA, PKCS1–V1_5.....	51
6.6.3.1.3	RSA, SSA, PSS.....	52
6.6.3.1.4	RSA, ISO9796–2, DS2, SIGN	53
6.6.3.1.5	RSASSA–PSS–SIGN	53
6.6.3.2	Signaturberechnung mittels ELC.....	54
6.6.4	Signaturprüfung.....	54
6.6.4.1	RSA, ISO9796–2, DS1, VERIFY, Option_RSA_CVC.....	54

6.6.4.2	Signaturprüfung mittels elliptischer Kurven	55
6.7	Vertraulichkeit von Daten, symmetrischer Fall	56
6.7.1	Symmetrische Verschlüsselung	56
6.7.1.1	Verschlüsselung 3DES, Option_DES	56
6.7.1.2	Verschlüsselung AES	57
6.7.2	Symmetrische Entschlüsselung	57
6.7.2.1	Entschlüsselung 3DES, Option_DES	57
6.7.2.2	Entschlüsselung AES	58
6.8	Vertraulichkeit von Daten, asymmetrischer Fall	59
6.8.1	Asymmetrische Verschlüsselung	59
6.8.1.1	RSA, ES, PKCS1 V1.5	59
6.8.1.2	RSA, OAEP, Verschlüsselung	59
6.8.1.3	Elliptic Curve Key Agreement	60
6.8.1.4	ELC Verschlüsselung	61
6.8.2	Asymmetrische Entschlüsselung	62
6.8.2.1	RSA, ES, PKCS1 V1.5, Decrypt	62
6.8.2.2	RSA, OAEP, Decrypt	63
6.8.2.3	Asymmetrische Entschlüsselung mittels ELC	64
7	CV-Zertifikat	65
7.1	CV-Zertifikat für RSA-Schlüssel, Option_RSA_CVC	65
7.1.1	Bestandteile eines CV-Zertifikats für RSA-Schlüssel	65
7.1.1.1	Certificate Profile Identifier (CPI)	65
7.1.1.2	Certification Authority Reference (CAR)	65
7.1.1.3	Certificate Holder Reference (CHR)	65
7.1.1.4	Certificate Holder Autorisation (CHA)	66
7.1.1.5	Object Identifier (OID)	66
7.1.1.6	Öffentlicher Schlüssel	66
7.1.1.6.1	Modulus	66
7.1.1.6.2	Öffentlicher Exponent	66
7.1.2	Zertifikatsprofile für RSA-Schlüssel	66
7.1.2.1	CV-Zertifikat für CA-Schlüssel	66
7.1.2.2	CV-Zertifikat für Authentisierungsschlüssel	67
7.1.3	Struktur und Inhalt eines CV-Zertifikats für RSA-Schlüssel	67
7.2	CV-Zertifikate für ELC-Schlüssel (informativ)	68
8	Objekte	69
8.1	Diverse Attribute (normativ)	69
8.1.1	File Identifier	69
8.1.2	Short File Identifier	69
8.1.3	Life Cycle Status	69
8.1.4	Zugriffsregelliste	70
8.1.5	Rekord	71
8.1.6	SE-Identifizier	72
8.1.7	PIN	72
8.1.8	Datum	72
8.2	Schlüsselmaterial (normativ)	73

8.2.1	Symmetrische Schlüssel.....	73
8.2.1.1	3DES-Schlüssel, Option_DES	73
8.2.1.2	AES-128-Schlüssel.....	73
8.2.1.3	AES-192-Schlüssel.....	73
8.2.1.4	AES-256-Schlüssel.....	74
8.2.2	Domainparameter für elliptische Kurven	74
8.2.3	Privater Schlüssel.....	75
8.2.3.1	Privater RSA-Schlüssel.....	75
8.2.3.2	Privater ELC-Schlüssel	76
8.2.4	Öffentlicher Schlüssel.....	76
8.2.4.1	Öffentlicher RSA-Schlüssel.....	76
8.2.4.2	Öffentlicher ELC-Schlüssel	76
8.2.5	Transportschutz für ein Passwort	76
8.3	File (normativ).....	77
8.3.1	Ordner	77
8.3.1.1	Applikation	79
8.3.1.2	Dedicated File.....	80
8.3.1.3	Application Dedicated File.....	80
8.3.2	Datei.....	80
8.3.2.1	Transparentes Elementary File	81
8.3.2.2	Strukturiertes Elementary File	82
8.3.2.2.1	Linear variables Elementary File.....	84
8.3.2.2.2	Linear fixes Elementary File	84
8.3.2.2.3	Zyklisches Elementary File	85
8.3.3	File Control Parameter.....	85
8.4	Reguläres-Passwort (normativ)	87
8.5	Multireferenz-Passwort (normativ)	91
8.6	Schlüsselobjekt (normativ)	92
8.6.1	Symmetrisches Authentisierungsobjekt	93
8.6.2	Symmetrisches Kartenverbindungsobjekt	95
8.6.3	Privates Schlüsselobjekt.....	97
8.6.4	Öffentliches Schlüsselobjekt.....	102
8.6.4.1	Öffentliches Signaturprüfobjekt	103
8.6.4.2	Öffentliches Authentisierungsobjekt.....	104
8.6.4.3	Öffentliches Verschlüsselungsobjekt.....	105
8.7	Datenobjekte (informativ).....	106
8.8	Security Environment (informativ)	106
8.9	Sicherheitsstatus (informativ)	108
9	Objektsystem (normativ).....	109
9.1	Aufbau und Strukturtiefe.....	109
9.2	Objektsuche	113
9.2.1	Filesuche	113
9.2.2	Passwortsuche	113
9.2.3	Suche nach einem Schlüsselobjekt	114

9.2.3.1	Suche nach einem geheimen Schlüsselobjekt mittels keyReference	115
9.2.3.2	Suche nach einem öffentlichen Schlüsselobjekt	116
9.3	Cache für öffentliche Schlüsselobjekte	117
10	Zugriffskontrolle (normativ)	120
10.1	Zugriffsart	120
10.2	Zugriffsbedingung	121
10.3	Zugriffsregel	123
10.4	Zugriffsregelauswertung	124
10.5	Auslesen von Zugriffsregeln	125
11	Kommunikation (normativ)	126
11.1	Request – Response	126
11.2	Elektrische Schnittstellen	126
11.2.1	Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11]	126
11.2.2	Übertragungsprotokoll gemäß [ISO/IEC 7816-12]	129
11.2.3	Kontaktlose Datenübertragung gemäß ISO/IEC 14443	129
11.3	OSI-Referenzmodell (informativ)	130
11.4	Kommandobearbeitung	131
11.5	Kommando-APDU	132
11.5.1	Class Byte	132
11.5.2	Instruction Byte	133
11.5.3	Parameter P1	133
11.5.4	Parameter P2	133
11.5.5	Datenfeld	133
11.5.6	LeFeld	134
11.6	Antwort-APDU	135
11.6.1	Datenfeld	135
11.6.2	Trailer	135
11.7	Zulässige Kommando-Antwort-Paare	136
11.7.1	Case 1 Kommando-Antwort-Paar	136
11.7.2	Case 2 Kommando-Antwort-Paar	136
11.7.2.1	Case 2 Short Kommando	136
11.7.2.2	Case 2 Extended Kommando	137
11.7.2.3	Case 2 Response	138
11.7.3	Case 3 Kommando	138
11.7.3.1	Case 3 Short Kommando	138
11.7.3.2	Case 3 Extended Kommando	139
11.7.3.3	Case 3 Response	139
11.7.4	Case 4 Kommando	140
11.7.4.1	Case 4 Short Kommando	140
11.7.4.2	Case 4 Extended Kommando	141
11.7.4.3	Case 4 Response	141
11.8	Command Chaining	142

11.9	Längenbeschränkung von APDU	143
12	Kanalkontext (normativ)	145
12.1	Attribute eines logischen Kanals	145
12.2	Reset-Verhalten	148
12.3	Setzen eines Sicherheitsstatus	149
12.4	Löschen eines Sicherheitsstatus	150
12.4.1	Löschen des Sicherheitszustandes eines Schlüssels	150
12.4.2	Löschen der Sicherheitszustände eines Ordners	151
12.4.3	Löschen von Sessionkeys	151
12.5	Setzen eines Passwortstatus	152
12.6	Löschen eines Passwortstatus	152
13	Gesicherte Kommunikation (normativ)	154
13.1	Secure Messaging Layer	154
13.1.1	Ableitung von Sessionkeys	154
13.1.2	Bearbeitung einer Kommando-APDU	156
13.2	Sicherung einer Kommando-APDU	158
13.3	Sicherung einer Antwort-APDU	161
14	Kommandos (normativ)	164
14.1	Roll-Verhalten	165
14.1.1	Roll-Back	165
14.1.2	Roll-Forward	166
14.2	Management des Objektsystems	166
14.2.1	ACTIVATE	166
14.2.1.1	Use Case Aktivieren eines Ordners oder einer Datei	167
14.2.1.2	Use Case Aktivieren eines privaten oder symmetrischen Schlüsselobjektes	167
14.2.1.3	Use Case Aktivieren eines öffentlichen Schlüsselobjektes	167
14.2.1.4	Use Case Aktivieren eines Passwortobjektes	168
14.2.1.5	Antwort der Karte auf Aktivieren eines Files	169
14.2.1.6	Kommandoabarbeitung innerhalb der Karte	169
14.2.2	CREATE	170
14.2.3	DEACTIVATE	171
14.2.3.1	Use Case Deaktivieren eines Ordners oder einer Datei	171
14.2.3.2	Use Case Deaktivieren eines privaten oder symmetrischen Schlüsselobjektes	171
14.2.3.3	Use Case Deaktivieren eines öffentlichen Schlüsselobjektes	172
14.2.3.4	Use Case Deaktivieren eines Passwortobjektes	172
14.2.3.5	Antwort der Karte auf Deaktivieren eines Files	173
14.2.3.6	Kommandoabarbeitung innerhalb der Karte	173
14.2.4	Delete	175
14.2.4.1	Use Case Löschen eines Ordners oder einer Datei	175
14.2.4.2	Use Case Löschen eines privaten oder symmetrischen Schlüsselobjektes	175

14.2.4.3	Use Case Löschen eines öffentlichen Schlüsselobjektes	176
14.2.4.4	Use Case Löschen eines Passwortobjektes	176
14.2.4.5	Antwort der Karte auf Löschen eines Files	177
14.2.4.6	Kommandoabarbeitung innerhalb der Karte	177
14.2.5	LOAD APPLICATION	179
14.2.5.1	Use Case Anlegen neues Objekt, nicht Ende der Kommandokette	180
14.2.5.2	Use Case Anlegen neues Objekt, Ende der Kommandokette	180
14.2.5.3	Antwort der Karte auf Anlegen neues Objekt	181
14.2.5.4	Kommandoabarbeitung innerhalb der Karte	181
14.2.6	SELECT	184
14.2.6.1	Use Case Selektieren ohne AID, first, keine Antwortdaten	184
14.2.6.2	Use Case Selektieren ohne AID, first, Antwortdaten mit FCP	185
14.2.6.3	Use Case Selektieren ohne AID, next, keine Antwortdaten	185
14.2.6.4	Use Case Selektieren ohne AID, next, Antwortdaten mit FCP	186
14.2.6.5	Use Case Selektieren per AID, first, keine Antwortdaten	187
14.2.6.6	Use Case Selektieren per AID, first, Antwortdaten mit FCP	187
14.2.6.7	Use Case Selektieren per AID, next, keine Antwortdaten	188
14.2.6.8	Use Case Selektieren per AID, next, Antwortdaten mit FCP	189
14.2.6.9	Use Case Selektieren, DF oder ADF, keine Antwortdaten	190
14.2.6.10	Use Case Selektieren, DF oder ADF, Antwortdaten mit FCP	190
14.2.6.11	Use Case Selektieren übergeordnetes Verzeichnis ohne FCP	191
14.2.6.12	Use Case Selektieren übergeordnetes Verzeichnis mit FCP	192
14.2.6.13	Use Case Selektieren einer Datei, keine Antwortdaten	192
14.2.6.14	Use Case Selektieren einer Datei, Antwortdaten mit FCP	193
14.2.6.15	Zusammenfassung der SELECT-Kommando-Varianten	194
14.2.6.16	Antwort der Karte auf Selektieren eines Files	195
14.2.6.17	Kommandoabarbeitung innerhalb der Karte	195
14.2.7	TERMINATE CARD USAGE	198
14.2.7.1	Use Case Terminieren der Karte	198
14.2.7.2	Antwort der Karte auf terminieren der Karte	198
14.2.7.3	Kommandoabarbeitung innerhalb der Karte	199
14.2.8	TERMINATE DF	200
14.2.8.1	Use Case Terminieren eines Ordners	200
14.2.8.2	Antwort der Karte auf terminieren der Karte	200
14.2.8.3	Kommandoabarbeitung innerhalb der Karte	201
14.2.9	TERMINATE	202
14.2.9.1	Use Case Terminieren einer Datei	202
14.2.9.2	Use Case Terminieren eines privaten oder symmetrischen Schlüsselobjektes	202
14.2.9.3	Use Case Terminieren eines öffentlichen Schlüsselobjektes	203
14.2.9.4	Use Case Terminieren eines Passwortobjektes	203
14.2.9.5	Antwort der Karte auf terminieren von Datei, Schlüssel- oder Passwortobjekt	204
14.2.9.6	Kommandoabarbeitung innerhalb der Karte	204
14.3	Zugriff auf Daten in transparenten EF	206
14.3.1	ERASE BINARY	206
14.3.1.1	Use Case Löschen ohne shortFileIdentifier in transparenten EF	206
14.3.1.2	Use Case Löschen mit shortFileIdentifier in transparenten EF	207
14.3.1.3	Antwort der Karte auf Löschen in transparenten EF	207
14.3.1.4	Kommandoabarbeitung innerhalb der Karte	208
14.3.2	READ BINARY	209

14.3.2.1	Use Case Lesen ohne shortFileIdentifier in transparenten EF	210
14.3.2.2	Use Case Lesen mit shortFileIdentifier in transparenten EF.....	210
14.3.2.3	Antwort der Karte auf Lesen in transparenten EF	211
14.3.2.4	Kommandoabarbeitung innerhalb der Karte.....	211
14.3.3	SEARCH BINARY	213
14.3.4	SET LOGICAL EOF	213
14.3.4.1	Use Case Setzen logical EOF ohne shortFileIdentifier.....	213
14.3.4.2	Use Case Setzen logical EOF mit shortFileIdentifier.....	214
14.3.4.3	Antwort der Karte auf Setzen logical EOF in transparenten EF.....	214
14.3.4.4	Kommandoabarbeitung innerhalb der Karte.....	215
14.3.5	UPDATE BINARY	217
14.3.5.1	Use Case Schreiben ohne shortFileIdentifier in transparenten EF.....	217
14.3.5.2	Use Case Schreiben mit shortFileIdentifier in transparenten EF	218
14.3.5.3	Antwort der Karte auf Schreiben in transparenten EF	218
14.3.5.4	Kommandoabarbeitung innerhalb der Karte.....	219
14.3.6	WRITE BINARY	221
14.3.6.1	Use Case Anfügen ohne shortFileIdentifier in transparenten EF.....	221
14.3.6.2	Use Case Anfügen mit shortFileIdentifier in transparenten EF.....	221
14.3.6.3	Antwort der Karte auf Anfügen in transparenten EF.....	222
14.3.6.4	Kommandoabarbeitung innerhalb der Karte.....	222
14.4	Zugriff auf strukturierte Daten	224
14.4.1	ACTIVATE RECORD	225
14.4.1.1	Use Case Aktivieren eines Rekords ohne shortFileIdentifier	225
14.4.1.2	Use Case Aktivieren eines Rekords mit shortFileIdentifier	225
14.4.1.3	Use Case Aktivieren aller Rekords ab P1 ohne shortFileIdentifier.....	226
14.4.1.4	Use Case Aktivieren aller Rekords ab P1 mit shortFileIdentifier.....	226
14.4.1.5	Antwort der Karte auf Aktivieren eines Rekords.....	227
14.4.1.6	Kommandoabarbeitung innerhalb der Karte.....	227
14.4.2	APPEND RECORD	229
14.4.2.1	Use Case Anlegen neuer Rekord, ohne shortFileIdentifier.....	229
14.4.2.2	Use Case Anlegen neuer Rekords, mit shortFileIdentifier	230
14.4.2.3	Antwort der Karte auf Anlegen eines neuen Rekords.....	231
14.4.2.4	Kommandoabarbeitung innerhalb der Karte.....	231
14.4.3	DEACTIVATE RECORD.....	234
14.4.3.1	Use Case Deaktivieren eines Rekords ohne shortFileIdentifier.....	234
14.4.3.2	Use Case Deaktivieren eines Rekords mit shortFileIdentifier.....	234
14.4.3.3	Use Case Deaktivieren aller Rekords ab P1 ohne shortFileIdentifier.....	235
14.4.3.4	Use Case Deaktivieren aller Rekords ab P1 mit shortFileIdentifier.....	236
14.4.3.5	Antwort der Karte auf Deaktivieren eines Rekords.....	236
14.4.3.6	Kommandoabarbeitung innerhalb der Karte.....	237
14.4.4	DELETE RECORD.....	238
14.4.4.1	Use Case Löschen eines Rekords ohne shortFileIdentifier	239
14.4.4.2	Use Case Löschen eines Rekords mit shortFileIdentifier	239
14.4.4.3	Antwort der Karte auf Entfernen eines Rekords	240
14.4.4.4	Kommandoabarbeitung innerhalb der Karte.....	240
14.4.5	ERASE RECORD	242
14.4.5.1	Use Case Löschen eines Rekordinhaltes ohne shortFileIdentifier.....	242
14.4.5.2	Use Case Löschen eines Rekordinhaltes mit shortFileIdentifier.....	243
14.4.5.3	Antwort der Karte auf Löschen des Inhaltes eines Rekords	243
14.4.5.4	Kommandoabarbeitung innerhalb der Karte.....	244
14.4.6	READ RECORD	246

14.4.6.1	Use Case Lesen ohne shortFileIdentifier in strukturierten EF.....	246
14.4.6.2	Use Case Lesen mit shortFileIdentifier in strukturierten EF.....	246
14.4.6.3	Antwort der Karte auf Lesen in strukturierten EF.....	247
14.4.6.4	Kommandoabarbeitung innerhalb der Karte.....	248
14.4.7	SEARCH RECORD	249
14.4.7.1	Use Case Suchen ohne shortFileIdentifier in strukturierten EF	250
14.4.7.2	Use Case Suchen mit shortFileIdentifier in strukturierten EF	250
14.4.7.3	Antwort der Karte auf Suchen in strukturierten EF	251
14.4.7.4	Kommandoabarbeitung innerhalb der Karte.....	252
14.4.8	UPDATE RECORD	253
14.4.8.1	Use Case Rekordinhalt schreiben, ohne shortFileIdentifier	254
14.4.8.2	Use Case Rekordinhalt schreiben, mit shortFileIdentifier	254
14.4.8.3	Antwort der Karte auf Schreiben in strukturierten EF	255
14.4.8.4	Kommandoabarbeitung innerhalb der Karte.....	255
14.4.9	WRITE RECORD.....	258
14.5	Zugriff auf Datenobjekte.....	258
14.5.1	GET DATA.....	258
14.5.2	PUT DATA	259
14.6	Benutzerverifikation	259
14.6.1	CHANGE REFERENCE DATA	259
14.6.1.1	Use Case Ändern eines Benutzergeheimnisses	259
14.6.1.2	Use Case Setzen eines Benutzergeheimnisses.....	260
14.6.1.3	Antwort der Karte auf Ändern eines Benutzergeheimnisses.....	261
14.6.1.4	Kommandoabarbeitung innerhalb der Karte.....	261
14.6.2	DISABLE VERIFICATION REQUIREMENT.....	263
14.6.2.1	Use Case Abschalten der Benutzerverifikation mit Benutzergeheimnis 263	
14.6.2.2	Use Case Abschalten der Benutzerverifikation ohne Benutzergeheimnis 264	
14.6.2.3	Antwort der Karte auf Abschalten der Benutzerverifikation.....	264
14.6.2.4	Kommandoabarbeitung innerhalb der Karte.....	265
14.6.3	ENABLE VERIFICATION REQUIREMENT	266
14.6.3.1	Use Case Einschalten der Benutzerverifikation mit Benutzergeheimnis 266	
14.6.3.2	Use Case Einschalten der Benutzerverifikation ohne Benutzergeheimnis 267	
14.6.3.3	Antwort der Karte auf Einschalten der Benutzerverifikation.....	267
14.6.3.4	Kommandoabarbeitung innerhalb der Karte.....	268
14.6.4	GET PIN STATUS	269
14.6.4.1	Use Case Auslesen des Status eines Passwortobjektes.....	270
14.6.4.2	Antwort der Karte auf Auslesen des PIN Status	270
14.6.4.3	Kommandoabarbeitung innerhalb der Karte.....	271
14.6.5	RESET RETRY COUNTER.....	272
14.6.5.1	Use Case Entsperren mit PUK, mit neuem Geheimnis	272
14.6.5.2	Use Case Entsperren mit PUK, ohne neues Geheimnis	273
14.6.5.3	Use Case Entsperren ohne PUK, mit neuem Geheimnis	273
14.6.5.4	Use Case Entsperren ohne PUK, ohne neues Geheimnis	274
14.6.5.5	Antwort der Karte auf Entsperren eines Benutzergeheimnisses.....	274
14.6.5.6	Kommandoabarbeitung innerhalb der Karte.....	275
14.6.6	VERIFY	277

14.6.6.1	Use Case Vergleich eines Benutzergeheimnisses	277
14.6.6.2	Antwort der Karte auf Vergleich eines Benutzergeheimnisses	277
14.6.6.3	Kommandoabarbeitung innerhalb der Karte	278
14.7	Komponentenauthentisierung	279
14.7.1	EXTERNAL AUTHENTICATE / MUTUAL AUTHENTICATE	279
14.7.1.1	Use Case externe Authentisierung ohne Antwortdaten	279
14.7.1.2	Use Case externe Authentisierung mit Antwortdaten	280
14.7.1.3	Antwort der Karte auf externe Authentisierung	281
14.7.1.4	Kommandoabarbeitung innerhalb der Karte	282
14.7.2	GENERAL AUTHENTICATE	287
14.7.2.1	Gegenseitige Authentisierung mittels PACE für Endnutzerkarten	288
14.7.2.1.1	Use Case PACE für Endnutzerkarten, Schritt 1a	288
14.7.2.1.2	Use Case PACE für Endnutzerkarten, Schritt 2a	288
14.7.2.1.3	Use Case PACE für Endnutzerkarten, Schritt 3a	289
14.7.2.1.4	Use Case PACE für Endnutzerkarten, Schritt 4a	290
14.7.2.2	Gegenseitige Authentisierung mittels ELC Schlüsseln	290
14.7.2.2.1	Use Case gegenseitige ELC-Authentisierung, Schritt 1	290
14.7.2.2.2	Use Case gegenseitige ELC-Authentisierung, Schritt 2	291
14.7.2.3	Authentisierung für asynchrone, symmetrische Kartenadministration	292
14.7.2.3.1	Use Case Authentisierung für asynchrone, sym. Administration, Schritt 1	292
14.7.2.3.2	Use Case Authentisierung für asynchrone, sym. Administration, Schritt 2	292
14.7.2.4	Gegenseitige Authentisierung mittels PACE für Sicherheitsmodule	293
14.7.2.4.1	Use Case PACE für Sicherheitsmodule, Schritt 1b	293
14.7.2.4.2	Use Case PACE für Sicherheitsmodule, Schritt 2b	294
14.7.2.4.3	Use Case PACE für Sicherheitsmodule, Schritt 3b	294
14.7.2.4.4	Use Case PACE für Sicherheitsmodule, Schritt 4b	295
14.7.2.4.5	Use Case PACE für Sicherheitsmodule, Schritt 5b	295
14.7.2.5	Authentisierung für asynchrone, asymmetrische Kartenadministration	296
14.7.2.5.1	Use Case Authentisierung für asynchrone, asym. Administration, Schritt 1	296
14.7.2.5.2	Use Case Authentisierung für asynchrone, asym. Administration, Schritt 2	296
14.7.2.6	Antwort der Karte auf generelle Authentisierung	297
14.7.2.7	Kommandoabarbeitung innerhalb der Karte	297
14.7.3	GET SECURITY STATUS KEY	308
14.7.3.1	Use Case Auslesen Sicherheitsstatus symmetrischer Schlüssels, Option_DES	309
14.7.3.2	Use Case Auslesen des Sicherheitsstatus einer Rolle, Option_RSA_CVC	309
14.7.3.3	Use Case Auslesen des Sicherheitsstatus einer Bitliste	309

14.7.3.4	Antwort der Karte auf Auslesen Sicherheitsstatus eines Schlüssels ..	310
14.7.3.5	Kommandoabarbeitung innerhalb der Karte.....	310
14.7.4	INTERNAL AUTHENTICATE	311
14.7.4.1	Use Case interne Authentisierung.....	312
14.7.4.2	Antwort der Karte auf interne Authentisierung.....	313
14.7.4.3	Kommandoabarbeitung innerhalb der Karte.....	313
14.8	Kryptoboxkommandos.....	316
14.8.1	PSO Compute Cryptographic Checksum	317
14.8.1.1	Use Case Berechnen einer kryptographischen Checksumme.....	317
14.8.1.2	Antwort der Karte auf Berechnen einer kryptographischen Checksumme 318	
14.8.1.3	Kommandoabarbeitung innerhalb der Karte.....	318
14.8.2	PSO Compute Digital Signature.....	319
14.8.2.1	Use Case Signieren des Datenfeldes, ohne „message recovery“.....	319
14.8.2.2	Use Case Signieren des Datenfeldes, mit „message recovery“.....	320
14.8.2.3	Antwort der Karte auf Signieren von Daten	321
14.8.2.4	Kommandoabarbeitung innerhalb der Karte.....	321
14.8.3	PSO Decipher.....	323
14.8.3.1	Use Case Entschlüsseln mittels RSA.....	323
14.8.3.2	Use Case Entschlüsseln mittels ELC.....	324
14.8.3.3	Use Case Entschlüsseln mittels symmetrischer Schlüssel.....	325
14.8.3.4	Antwort der Karte auf Entschlüsseln von Daten	325
14.8.3.5	Kommandoabarbeitung innerhalb der Karte.....	326
14.8.4	PSO Encipher.....	328
14.8.4.1	Use Case Verschlüsseln von Daten mittels übergebenem RSA- Schlüssel 328	
14.8.4.2	Use Case Verschlüsseln von Daten mittels übergebenem ELC- Schlüssel 329	
14.8.4.3	Use Case Verschlüsseln mittels gespeichertem RSA-Schlüssel	330
14.8.4.4	Use Case Verschlüsseln mittels gespeichertem ELC-Schlüssel.....	331
14.8.4.5	Use Case Verschlüsseln mittels symmetrischem Schlüssel.....	331
14.8.4.6	Antwort der Karte auf Verschlüsseln von Daten	332
14.8.4.7	Kommandoabarbeitung innerhalb der Karte.....	333
14.8.5	PSO Hash.....	335
14.8.6	PSO Transcipher	335
14.8.6.1	Use Case Umschlüsseln von Daten mittels RSA-Schlüssel	336
14.8.6.2	Use Case Umschlüsseln von Daten von RSA-Schlüssel nach ELC- Schlüssel 336	
14.8.6.3	Use Case Umschlüsseln von Daten mittels ELC.....	337
14.8.6.4	Use Case Umschlüsseln von Daten von ELC-Schlüssel nach RSA- Schlüssel 339	
14.8.6.5	Antwort der Karte auf Umschlüsseln von Daten	340
14.8.6.6	Kommandoabarbeitung innerhalb der Karte.....	340
14.8.7	PSO Verify Certificate	343
14.8.7.1	Use Case Import RSA-Schlüssels mittels Zertifikat, Option_RSA_CVC 343	
14.8.7.2	Use Case Import ELC-Schlüssels mittels Zertifikat	344
14.8.7.3	Antwort der Karte auf Vergleich eines Benutzergeheimnisses	345
14.8.7.4	Kommandoabarbeitung innerhalb der Karte.....	345
14.8.8	PSO Verify Cryptographic Checksum	350
14.8.8.1	Use Case Prüfung einer kryptographischen Checksumme	350

14.8.8.2	Antwort der Karte auf Berechnen einer kryptographischen Checksumme	350
14.8.8.3	Kommandoabarbeitung innerhalb der Karte	351
14.8.9	PSO Verify Digital Signature	352
14.8.9.1	Use Case Prüfen einer ELC-Signatur	352
14.8.9.2	Antwort der Karte auf Prüfen einer digitalen Signatur	353
14.8.9.3	Kommandoabarbeitung innerhalb der Karte	354
14.9	Verschiedenes	354
14.9.1	ENVELOPE	354
14.9.2	FINGERPRINT	354
14.9.2.1	Use Case Fingerprint über das COS berechnen	355
14.9.2.2	Antwort der Karte auf Fingerprintberechnung	355
14.9.2.3	Kommandoabarbeitung innerhalb der Karte	356
14.9.3	GENERATE ASYMMETRIC KEY PAIR	356
14.9.3.1	Use Case Generierung, ohne Überschreiben, ohne Referenz, ohne Ausgabe	357
14.9.3.2	Use Case Generierung, ohne Überschreiben, mit Referenz, ohne Ausgabe	357
14.9.3.3	Use Case Generierung, ggf. Überschreiben, ohne Referenz, ohne Ausgabe	358
14.9.3.4	Use Case Generierung, ggf. Überschreiben, mit Referenz, ohne Ausgabe	358
14.9.3.5	Use Case Auslesen vorhandener Schlüssel, ohne Referenz	359
14.9.3.6	Use Case Auslesen vorhandener Schlüssel, mit Referenz	359
14.9.3.7	Use Case Generierung, ohne Überschreiben, ohne Referenz, mit Ausgabe	360
14.9.3.8	Use Case Generierung, ohne Überschreiben, mit Referenz, mit Ausgabe	360
14.9.3.9	Use Case Generierung, ggf. Überschreiben, ohne Referenz, mit Ausgabe	361
14.9.3.10	Use Case Generierung, ggf. Überschreiben, mit Referenz, mit Ausgabe	362
14.9.3.11	Zusammenfassung der GENERATE ASYMMETRIC KEY PAIR-Kommando-Varianten	362
14.9.3.12	Antwort der Karte auf Schlüsselgenerierung	363
14.9.3.13	Kommandoabarbeitung innerhalb der Karte	363
14.9.4	GET CHALLENGE	365
14.9.4.1	Use Case Zufallszahl für DES oder RSA Authentisierung	366
14.9.4.2	Use Case Zufallszahl für AES oder ELC Authentisierung	366
14.9.4.3	Antwort der Karte auf Erzeugen einer Zufallszahl	366
14.9.4.4	Kommandoabarbeitung innerhalb der Karte	367
14.9.5	GET RANDOM	367
14.9.5.1	Use Case Erzeugen kryptographisch sicherer Zufallszahl	367
14.9.5.2	Antwort der Karte Erzeugen kryptographisch sichere Zufallszahl	368
14.9.5.3	Kommandoabarbeitung innerhalb der Karte	368
14.9.6	GET RESPONSE	369
14.9.7	LIST PUBLIC KEY	369
14.9.7.1	Use Case Auslesen der Liste öffentlicher Schlüsselobjekte	369
14.9.7.2	Antwort der Karte auf Auslesen einer Schlüsselliste	370
14.9.7.3	Kommandoabarbeitung innerhalb der Karte	370
14.9.8	MANAGE CHANNEL	372

14.9.8.1	Use Case Öffnen eines logischen Kanals	372
14.9.8.2	Use Case Schließen eines logischen Kanals	373
14.9.8.3	Use Case Zurücksetzen eines logischen Kanals.....	373
14.9.8.4	Use Case logischer Reset der Applikationsebene.....	374
14.9.8.5	Antwort der Karte auf Kanalmanagementoperationen.....	374
14.9.8.6	Kommandoabarbeitung innerhalb der Karte.....	374
14.9.9	MANAGE SECURITY ENVIRONMENT.....	376
14.9.9.1	Use Case Ändern des SE-Identifiers.....	376
14.9.9.2	Use Case Schlüsselauswahl zur internen, symmetrischen Authentisierung.....	377
14.9.9.3	Use Case Schlüsselauswahl zur internen, asymmetrischen Authentisierung.....	378
14.9.9.4	Use Case Schlüsselauswahl zur externen, symmetrischen Authentisierung.....	378
14.9.9.5	Use Case Schlüsselauswahl zur externen, asymmetrischen Authentisierung.....	379
14.9.9.6	Use Case Schlüsselauswahl zur symmetrischen, gegenseitigen Authentisierung.....	380
14.9.9.7	Use Case Schlüsselauswahl zur sym. Kartenverbindung ohne Kurvenangabe.....	381
14.9.9.8	Use Case Schlüsselauswahl zur sym. Kartenverbindung mit Kurvenangabe.....	382
14.9.9.9	Use Case Schlüsselauswahl für Signierschlüssel	383
14.9.9.10	Use Case Schlüsselauswahl zum Prüfen von CV-Zertifikaten	384
14.9.9.11	Use Case Schlüsselauswahl zur Datenent- oder Datenumschlüsselung.....	384
14.9.9.12	Use Case Schlüsselauswahl für Verschlüsselung	385
14.9.9.13	Antwort der Karte auf Management des Security Environments.....	386
14.9.9.14	Kommandoabarbeitung innerhalb der Karte.....	386
15	Authentisierungsprotokolle (normativ)	389
15.1	Externe Authentisierung	390
15.1.1	Externe Authentisierung mittels symmetrischer Schlüssel	390
15.1.2	RSA, asymmetrische Rollenauthentisierung, Option_RSA_CVC	391
15.1.3	ELC, asymmetrische Berechtigungsnachweis	391
15.2	Interne Authentisierung	391
15.3	Card-2-Card-Authentisierung ohne Sessionkey-Aushandlung	392
15.4	Aushandlung von Sessionkey	392
15.4.1	Sessionkeys mittels symmetrischer Authentisierungsobjekte.....	392
15.4.2	Sessionkeys mittels symmetrischer Kartenverbindungsobjekte	393
15.4.3	Sessionkeyaushandlung mittels RSA-Schlüssel, Option_DES.....	395
15.4.4	Sessionkeys mittels ELC-Schlüssel	396
15.5	Statisches Secure Messaging.....	399
16	Verschiedenes (normativ).....	401
16.1	Identifizierung	401
16.2	Codierungen für Trailer	404

Anhang A – Hinweise zur Sicherheitsevaluierung (informativ).....	406
Anhang B – Vorgaben zur Performanz.....	407
B.1 Einführung (informativ).....	407
B.2 Messaufbau (normativ).....	407
B.3 Anforderungen an die Steuersoftware (normativ).....	408
B.4 Anforderungen an das Interface Device (IFD) (normativ)	408
B.4.1 Anforderungen an das IFD bezüglich T=1	408
B.4.2 Anforderungen an das IFD für [ISO/IEC 7816-12] Datenübertragung	409
B.4.3 Anforderungen an das IFD bezüglich kontaktloser Datenübertragung	409
B.5 Allgemeines (normativ)	409
B.5.1 Normale Zeitmessung.....	409
B.5.1.1 Normale Zeitmessung für das Übertragungsprotokoll T=1	409
B.5.1.2 Normale Zeitmessung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12].....	410
B.5.1.3 Normale Zeitmessung für die kontaktlose Datenübertragung.....	410
B.5.2 Reguläre Aktivierung der Smartcard	410
B.5.2.1 Reguläre Aktivierung für das Übertragungsprotokoll T=1	410
B.5.2.2 Reguläre Aktivierung für das Übertragungsprotokoll gemäß [ISO/IEC 7816-12].....	411
B.5.2.3 Reguläre Aktivierung für die kontaktlose Datenübertragung.....	411
B.5.3 Punkteermittlung.....	411
B.5.4 Gesamtbewertung	412
B.6 Übertragungsgeschwindigkeit.....	416
B.6.1 Übertragungsgeschwindigkeit für das Übertragungsprotokoll T=1	416
B.6.2 Übertragungsgeschwindigkeit für das Protokoll [ISO/IEC 7816-12].....	417
B.6.3 Übertragungsgeschwindigkeit für kontaktlose Datenübertragung	417
B.7 Startsequenz für das Übertragungsprotokoll T=1	417
B.8 Messverfahren für Einzelkommandos (normativ)	418
B.8.1 ACTIVATE, DEACTIVATE, DELETE, LOAD APPLICATION, TERMINATE	418
B.8.2 SELECT Datei	423
B.8.3 FINGERPRINT	424
B.8.4 TERMINATE CARD USAGE	424
B.8.5 SET LOGICAL EOF, WRITE BINARY	425
B.8.6 ERASE BINARY, UPDATE BINARY.....	426
B.8.7 READ BINARY.....	428
B.8.8 Rekord orientierte Kommandos	429
B.8.9 SEARCH RECORD.....	430
B.8.10 Symmetrische Sessionkeyaushandlung für Secure Messaging	432
B.8.11 Schlüsselimport und asymmetrische Authentisierungsprotokolle	433
B.8.11.1 ELC 256.....	433
B.8.11.2 ELC 384.....	435
B.8.11.3 ELC 512.....	437
B.8.11.4 RSA 2048	439
B.8.11.5 Testablauf Schlüsselimport und asymmetrische Authentisierung ...	439
B.8.12 INTERNAL AUTHENTICATE zur Rollenauthentisierung	442
B.8.13 PSO Compute Digital Signature mittels signPSS.....	443

B.8.14	Signaturerzeugung und –verifikation mittels signECDSA	444
B.8.15	PSO Encipher und PSO Decipher mittels rsaDecipherOaep	445
B.8.16	PSO Encipher und PSO Decipher mittels elcSharedSecretCalculation ..	446
B.8.17	Selektieren von Ordnern und Logical Channel Reset	448
B.8.18	MANAGE SECURITY ENVIRONMENT	449
B.8.19	GENERAL AUTHENTICATE, PACE	450
B.8.20	Symmetrische Sessionkeyaushandlung für Trusted Channel	450
B.8.21	Sessionkeynutzung im Trusted Channel	451
B.8.22	GET RANDOM	454
B.8.23	Öffnen und Schließen logischer Kanäle	455
B.9	Kartenkonfiguration für Performanztests (normativ)	456
B.9.1	Attribute des Objektsystems	456
B.9.1.1	Answer To Reset	456
B.9.2	Allgemeine Festlegungen zu Attributstabellen	457
B.9.3	Root, die Wurzelapplikation	458
B.9.3.1	/ MF / EF.ATR	458
B.9.3.2	/ MF / EF.DIR	459
B.9.3.3	/ MF / EF.GDO	459
B.9.4	Anwendung für Authentisierungsprotokolle, DF.Auth	460
B.9.4.1	/ MF / DF.Auth / PrK.Auth_ELC256	461
B.9.4.2	/ MF / DF.Auth / PrK.Auth_ELC384	461
B.9.4.3	/ MF / DF.Auth / PrK.Auth_ELC512	461
B.9.4.4	/ MF / DF.Auth / PrK.Auth_RSA2048	462
B.9.4.5	/ MF / DF.Auth / PuK.RCA_ELC256	462
B.9.4.6	/ MF / DF.Auth / PuK.RCA_ELC384	462
B.9.4.7	/ MF / DF.Auth / PuK.RCA_ELC512	463
B.9.4.8	/ MF / DF.Auth / PuK.RCA_RSA2048	463
B.9.4.9	/ MF / DF.Auth / SK.AES128	463
B.9.4.10	/ MF / DF.Auth / SK.AES192	464
B.9.4.11	/ MF / DF.Auth / SK.AES256	464
B.9.4.12	/ MF / DF.Auth / TC.AES128	464
B.9.4.13	/ MF / DF.Auth / TC.AES192	465
B.9.4.14	/ MF / DF.Auth / TC.AES256	465
B.9.5	Anwendung für IAS Services, DF.IAS	465
B.9.5.1	/ MF / DF.IAS / PrK.X509_ELC256	466
B.9.5.2	/ MF / DF.IAS / PrK.X509_ELC384	466
B.9.5.3	/ MF / DF.IAS / PrK.X509_ELC512	467
B.9.5.4	/ MF / DF.IAS / PrK.X509_RSA2048	467
B.9.5.5	/ MF / DF.IAS / PrK.X509_RSA3072	468
B.9.6	Anwendung für LCS Use Cases, DF.LCS	469
B.9.6.1	/ MF / DF.LCS / CAN_256	470
B.9.6.2	/ MF / DF.LCS / EF.LCS	470
B.9.6.3	/ MF / DF.LCS / PIN.LCS	470
B.9.6.4	/ MF / DF.LCS / PrK.LCS	471
B.9.6.5	/ MF / DF.LCS / PuK.LCS	471
B.9.6.6	/ MF / DF.LCS / SK.LCS	472
B.9.7	Anwendung für Select EF Use Cases, DF.SelectEF	472
B.9.7.1	/ MF / DF.SelectEF / EF.xx	472
B.9.8	Anwendung für rekordorientierte Dateioperationen, DF.strukturiert	473
B.9.8.1	/ MF / DF.strukturiert / EF.strukturiert	473
B.9.9	Anwendung für transparente Dateioperationen, DF.transparent	474

B.9.9.1	/ MF / DF.transparent / EF.transparent.....	474
Anhang C – Domainparameter elliptischer Kurven (informativ)		475
C.1	ansix9p256r1, OID = {1.2.840.10045.3.1.7} = '2A8648CE3D030107'	475
C.2	ansix9p384r1, OID = {1.3.132.0.34} = '2B81040022'	475
C.3	brainpoolP256r1, OID={1.3.36.3.3.2.8.1.1.7}='2B2403030208010107'	476
C.4	brainpoolP384r1, OID={1.3.36.3.3.2.8.1.1.11}='2B240303020801010B'	476
C.5	brainpoolP512r1, OID={1.3.36.3.3.2.8.1.1.13}='2B240303020801010D'	476
Anhang D – Erläuterungen zu Wertebereichen (informativ).....		477
Anhang E – Verzeichnisse (informativ)		478
E.1	Abkürzungen.....	478
E.2	Glossar	479
E.3	Abbildungsverzeichnis.....	479
E.4	Tabellenverzeichnis.....	479
E.5	Referenzierte Dokumente.....	488
E.5.1	Dokumente der gematik.....	488
E.5.2	Weitere Dokumente	489
Anhang F – Asynchrone, gesicherte APDU-Sequenzen (informativ)		492
F.1	Einleitung	492
F.2	CMS-Kommunikationsmuster.....	492
F.2.1	Synchrone Kommunikation zwischen CMS und Smartcard	492
F.2.2	Asynchrone Kommunikation zwischen CMS und Smartcard.....	492
F.3	Anforderungen an die asynchrone Kommunikation	493
F.4	Lösungskonzept	494
F.5	Kryptographie.....	496
F.5.1	CMS Aktivitäten	496
F.5.2	Puffer-Aktivitäten	497
F.5.3	Smartcard-Aktivitäten	497
F.6	Auswirkungen auf die Kommandoschnittstelle.....	498
Anhang G – CV-Zertifikate für RSA-Schlüssel, Option_RSA_CVC		499
Anhang H – CV-Zertifikate für ELC-Schlüssel (normativ).....		500
Anhang I – Speichern öffentlicher Schlüssel (informativ)		501
I.1	Definitionen.....	501
I.2	Perspektiven	502

1 Einordnung des Dokuments

1.1 Zielsetzung

Diese Spezifikation definiert die Anforderungen an die Funktionalität einer Betriebssystemplattform (COS 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 unterstützt werden müssen.

Der Aufbau dieses Dokumentes gliedert sich wie folgt:

- Kapitel 1 Einordnung des Dokuments enthält Aussagen zum Umgang mit diesem Dokument.
- Kapitel 2 Optionen wird in einer späteren Version Eingangsanforderungen enthalten, welche die Basis für die normativen Aussagen in späteren Kapiteln bilden werden.
- Kapitel 3 Systemüberblick (informativ) 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 (informativ) 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 (normativ) 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 (normativ) beschreibt, wie sich Informationen persistent auf einer Smartcard hierarchisch speichern lassen.
- Kapitel 10 Zugriffskontrolle (normativ) beschreibt den Schutz von Informationen auf einer Smartcard vor unberechtigtem Zugriff.
- Kapitel 11 Kommunikation (normativ) beschreibt, wie die Smartcard Informationen mit anderen Systemen austauscht.

- Kapitel 12 Kanalkontext (normativ) 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 (normativ) beschreibt, wie die Smartcard Informationen kryptographisch geschützt mit anderen Systemen austauscht.
- Kapitel 14 Kommandos (normativ) 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 (normativ) beschreibt Sequenzen, die aus mehr als einem Kommando bestehen.
- Kapitel 16 Verschiedenes (normativ) spezifiziert konkrete Werte für eine Reihe von Platzhaltern.

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, mit Kapitel 14 zu beginnen. Dort werden, wenn möglich, Verweise auf andere Kapitel gesetzt, wenn Dinge dort ausführlicher beschrieben werden. Wegen der besonderen Bedeutung des Kapitels 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, Kryptoboxkommandos 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 Schutzrechtinhabern 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 auf einer Versichertenkarte zu finden sind, ist nicht Gegenstand dieses Dokumentes. Diese finden sich 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: 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: 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 (1): Beispiel: 300 GB \approx 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. Falls das Attribut wieder ein Objekt mit weiteren Attributen ist, dann sind auch längere Bezeichnungen möglich.

G1	Abkürzung für Generation 1, bezeichnet die vorherige Version des Dokumentes, in der Regel ergänzt um den Zusatz „normativ“.
G2	Abkürzung für Generation 2, bezeichnet diese Version des Dokumentes, in der Regel ergänzt um den Zusatz „normativ“.
'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).

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 enthalten mag. 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 *seldentifizier* 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.
- In den 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 3: 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

1.5.5 Verwendung von Schlüsselworten

Anforderungen als Ausdruck normativer Festlegungen werden durch eine eindeutige ID in eckigen Klammern sowie die dem RFC 2119 [RFC2119] entsprechenden, in Großbuchstaben geschriebenen deutschen Schlüsselworte MUSS, DARF NICHT, SOLL, SOLL NICHT, KANN gekennzeichnet. Abwandlungen von „MUSS“ zu „MÜSSEN“ etc. sind der Grammatik geschuldet.

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 in diesem Dokument stattdessen *„Eine leere Liste DARF KEIN Element besitzen.“* verwendet.

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 einer Generation 2 Smartcard.

(N000.020) K_IC, Option_USB_Schnittstelle:

- a. Das IC und das COS einer Smartcard KÖNNEN die Option_USB_Schnittstelle unterstützen.
- b. Falls das IC und das COS einer Smartcard die Option_USB_Schnittstelle
 - 1. unterstützen, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_USB_Schnittstelle gekennzeichnet sind.
 - 2. nicht unterstützen, dann DÜRFEN mit Option_USB_Schnittstelle gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

(N000.022) K_IC, Option_kontaktlose_Schnittstelle:

- a. Das IC und das COS einer Smartcard KÖNNEN die Option_kontaktlose_Schnittstelle unterstützen.
- b. Falls das IC und das COS einer Smartcard die Option_kontaktlose_Schnittstelle
 - 1. unterstützen, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_kontaktlose_Schnittstelle gekennzeichnet sind.
 - 2. nicht unterstützt, dann DÜRFEN mit Option_kontaktlose_Schnittstelle gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

(N000.024) K_COS {K_Karte}, Option_logische_Kanäle:

- a. Das COS KANN die Option_logische_Kanäle unterstützen.
- b. Falls das COS die Option_logische_Kanäle
 - 1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_logische_Kanäle gekennzeichnet sind.
 - 2. nicht unterstützt, dann DÜRFEN mit Option_logische_Kanäle gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

(N000.026) K_COS {K_Karte}, Option_Kryptobox:

- a. Das COS KANN die Option_Kryptobox unterstützen.
- b. Falls das COS die Option_Kryptobox
 - 1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_Kryptobox gekennzeichnet sind.
 - 2. nicht unterstützt, dann DÜRFEN mit Option_Kryptobox gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

(N000.028) K_COS {K_Karte}, Option_PACE_PCD:

- a. Das COS KANN die Option_PACE_PCD unterstützen.

b. Falls das COS die Option_PACE_PCD

1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_PACE_PCD gekennzeichnet sind.
2. nicht unterstützt, dann DÜRFEN mit Option_PACE_PCD gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

(N000.030) K_COS {Karte}, Option_DES:

a. Das COS KANN die Option_DES unterstützen.

b. Falls das COS die Option_DES

1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_DES und Option_RSA_CVC gekennzeichnet sind.
2. nicht unterstützt, dann DÜRFEN mit Option_DES gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

Hinweis (2): 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.

(N000.032) K_C K_COS {Karte}, Option_RSA_CVC:

a. Das COS KANN die Option_RSA_CVC unterstützen.

b. Falls das COS die Option_RSA_CVC

1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_RSA_CVC gekennzeichnet sind.
2. nicht unterstützt, dann DÜRFEN mit Option_RSA_CVC gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

Hinweis (3): Die Option_RSA_CVC wird derzeit zur Implementierung folgende Karten benötigt:

- a. Karten der Generation 1, also eGK G1 und eGK G1+,
- b. HBA und SMC-B der Generation 2 und Generation G2.1.

(N000.034) K_COS {Karte}, Option_RSA_KeyGeneration:

a. Das COS KANN die Option_RSA_KeyGeneration unterstützen.

b. Falls das COS die Option_RSA_KeyGeneration

1. unterstützt, dann MÜSSEN zusätzlich zu allen nicht gekennzeichneten Anforderungen auch alle Anforderungen erfüllt werden, die mit Option_RSA_KeyGeneration gekennzeichnet sind.
2. nicht unterstützt, dann DÜRFEN mit Option_RSA_KeyGeneration gekennzeichneten Anforderung NICHT relevant für funktionale Tests sein.

Hinweis (4): Die Option_RSA_KeyGeneration wird derzeit lediglich im Rahmen der Personalisierung von Karten der Typen HBA und SMC-B 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 algld 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 14.2.7). Da nach Ausführung eines solchen Kommandos herstellerspezifisch noch gewisse Kommandos möglich sind (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:

Input:	<i>in</i>	Bitstring mit beliebigem Inhalt und beliebiger Länge
Output:	<i>out</i>	Integer, Anzahl der Bits aus denen <i>in</i> besteht
Errors:	–	Keine
Notation:		$out = \text{BitLength}(in)$

Beispiele:

- a. $\text{BitLength}(\text{''})} = 0$
- b. $\text{BitLength}(\text{'0'})} = 1$, $\text{BitLength}(\text{'1'})} = 1$,
- c. $\text{BitLength}(\text{'00'})} = 2$, $\text{BitLength}(\text{'01'})} = 2$,
- d. $\text{BitLength}(\text{'10'})} = 2$, $\text{BitLength}(\text{'11'})} = 2$, ...

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:

Input:	<i>in</i>	Oktettstring mit beliebigem Inhalt und beliebiger Länge, oder nicht-negative ganze Zahl
Output:	<i>out</i>	Integer, Anzahl der Oktette aus denen <i>in</i> besteht, oder Anzahl Oktette, die mindestens nötig sind, um eine nicht-negative ganze Zahl zu codieren.
Errors:	–	Keine
Notation:		$\text{out} = \text{OctetLength}(\text{in})$

Hinweis (5): Beispiele:

- a. $\text{OctetLength}(\text{''})} = 0$
- b. $\text{OctetLength}(\text{'0034'})} = 2$
- c. $\text{OctetLength}(0)} = 1$, weil die Zahl 0 in einem Oktett codierbar ist, $0 = \text{'00'}$.
- d. $\text{OctetLength}(127)} = 1$, weil $127 = \text{'7F'}$
- e. $\text{OctetLength}(255)} = 1$, weil $255 = \text{'FF'}$
- f. $\text{OctetLength}(256)} = 2$, weil $256 = \text{'0100'}$.

Hinweis (6): ACHTUNG: Der Oktettstring '0000FFFF' lässt sich als Repräsentant der Zahl 65535 interpretieren. Ohne führende Nullen lautet die hexadezimale Repräsentation der Zahl $65535 = \text{'FFFF'}$. Daraus folgt: $\text{OctetLength}(\text{'0000FFFF'}) = 4$, aber $\text{OctetLength}(65535) = 2$.

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:

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

Notation:	$out = I2OS(x, n)$
-----------	----------------------

*Hinweis (7): In gewissen Grenzen ist dies die Umkehrfunktion zu (N000.200).
(N000.100) K_COS*

Das Prinzip ist, die nicht-negative ganze Zahl x als Ziffernfolge zur Basis 256 zu notieren und dann nur die niedrigwertigsten Ziffern für die Ausgabe zu verwenden:

- a. Schritt 1: $x = 256^0 x_0 + 256^1 x_1 + 256^2 x_2 + \dots + 256^i x_i + \dots$
- b. Schritt 2: $M_i = x_i$.
Anmerkung: Jede Ziffer wird vorzeichenlos in einem Oktett codiert.
- c. Schritt 3: $out = M_{n-1} \parallel M_{n-2} \parallel \dots \parallel M_2 \parallel M_1 \parallel M_0$.

Hinweis (8): 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:

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

*Hinweis (9): In gewissen Grenzen ist dies die Umkehrfunktion zu (N000.100).
(N000.200) K_COS*

Das Prinzip ist, jedes Oktett als Ziffer zur Basis 256 einer nicht-negativen ganzen Zahl im Big-Endian-Format aufzufassen.

- a. Schritt 1: $n = \text{OctetLength}(in)$
- b. Falls n
 1. gleich null ist, dann ist $out = 0$.
 2. ungleich null ist, dann wähle out so, dass gilt $I2OS(out, n) = in$.

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:

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	Falls PO nicht im Format „uncompressed encoding“ vorliegt
Notation:		$P = OS2P(PO, dP)$

(N000.300) K_COS

Die Decodierung von PO erfolgt gemäß [BSI-TR-03111#3.2.1]:

- Schritt 1: Falls $\text{OctetLength}(PO)$ ungleich $(2 \cdot dP.L + 1)$ ist, dann gebe den Fehler „ERROR“ zurück und beende diesen Algorithmus.
- Schritt 2: Teile PO auf gemäß:

$$PO = PC \parallel X \parallel Y,$$

$$1 = \text{OctetLength}(PC),$$

$$dP.L = \text{OctetLength}(X) = \text{OctetLength}(Y).$$
- Schritt 3: Falls PC ungleich '04' ist, dann gebe den Fehler „ERROR“ zurück und beende diesen Algorithmus.
- Schritt 4: $P = (x, y) = (OS2I(X) \bmod dP.p, OS2I(Y) \bmod dP.p).$
- Schritt 5: Falls 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:

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

(N000.400) K_COS

Die Codierung von P erfolgt gemäß [BSI-TR-03111#3.2.1]:

$$PO = '04' \parallel I2OS(x, n) \parallel I2OS(y, n).$$

5.7 Extrahiere führende Elemente

5.7.1 Extrahiere führende Bits

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

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

(N000.500) K_COS

Es gilt die Vorbedingung *n* kleiner gleich *s*.

(N000.600) K_COS

Der Bitstring *out* enthält die *n* MSBit von *in*.

5.7.2 Extrahiere führende Oktette

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

Input:	<i>in</i>	Oktettstring der Länge <i>s</i> Oktett
	<i>n</i>	Integer, Anzahl der zu extrahierenden Elemente
Output:	<i>out</i>	Oktettstring der Länge <i>n</i> Oktett
Errors:	–	Keine
Notation:		<i>out</i> = Extract_MSByte(<i>in</i> , <i>n</i>)

(N000.700) K_COS

Es gilt die Vorbedingung *n* kleiner gleich *s*.

(N000.800) K_COS

Der Oktettstring *out* enthält die *n* MSByte von *in*.

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:

Input:	<i>in</i>	Oktettstring mit beliebigem Inhalt und beliebiger Länge
	<i>n</i>	Integer, gibt die Blocklänge von <i>out</i> in Oktett an
Output:	<i>out</i>	Oktettstring mit einer Anzahl Oktette, die Vielfaches von <i>n</i> ist
Errors:	–	Keine
Notation:		$out = \text{PaddingIso}(in, n)$

Hinweis (10): Dies ist die Umkehrfunktion zu (N001.000).

(N000.900) K_COS

Führe folgende Aktion durch:

- Schritt 1: $out = in \parallel '80'$.
- Schritt 2: Falls $0 = \text{OctetLength}(out) \bmod n$, dann gebe *out* zurück und beende den Algorithmus, sonst fahre mit Schritt 3 fort.
- Schritt 3: $out = out \parallel '00'$.
- Schritt 4: Fahre mit Schritt 2 fort.

5.9 Truncatelso

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:

Input:	<i>in</i>	Oktettstring mit beliebigem Inhalt und einer Anzahl Oktette, die Vielfaches von <i>n</i> ist
	<i>n</i>	Integer, gibt die Blocklänge in Oktett an, auf die gepadded wurde
Output:	<i>out</i>	Oktettstring mit beliebigem Inhalt und beliebiger Länge
Errors:	<i>paddingError</i>	Die Länge von <i>in</i> ist kein Vielfaches von <i>n</i> Es sind zu viele Paddingbits vorhanden
Notation:		$out = \text{Truncatelso}(in, n)$

Hinweis (11): Dies ist die Umkehrfunktion (N000.900).

(N001.000) K_COS

Führe folgende Aktion durch:

- Schritt 1: $len = \text{OctetLength}(in)$
Falls *len* kein Vielfaches von *n* ist, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
- Schritt 2: Falls *len* gleich null ist, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.
- Schritt 3: Falls LSByte von *in* den Wert '00' hat, dann setze
 $in = \text{Extract_MSByte}(in, \text{OctetLength}(in) - 1)$
und fahre mit Schritt 2 fort.
- Schritt 4: Falls LSByte von *in* nicht den Wert '80' hat, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.

- e. Schritt 5: $out = \text{Extract_MSByte}(in, \text{OctetLength}(in) - 1)$
- f. Schritt 6: Falls $(len - \text{OctetLength}(out))$ größer als n ist, dann breche diesen Algorithmus mit der Fehlermeldung *paddingError* ab.

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:

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. <i>Hinweis (12): Dieser Fehler ist für Smartcards derzeit nicht praxisrelevant, da sie erst dann auftreten, wenn Z oder N oberhalb mehrerer Gigabyte liegen.</i>
Notation:		$N = \text{MGF}(Z, L_N, i)$

Hinweis (13): Wenn diese Funktion mit $i = 0$ aufgerufen wird, dann ist das Ergebnis konform zu

- a. [ISO/IEC 9796-2#C], und
- b. [PKCS#1#B.2.1].

Hinweis (14): Wenn diese Funktion mit $i = 1$ 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.

(N001.100) K_COS

Führe folgende Aktionen durch:

- a. Schritt 1: Setze $n = ''$, (Anmerkung: Leerer Oktettstring).
- b. Schritt 2: Setze $n = n \parallel \text{SHA_256}(\text{BS2OS}(Z) \parallel \text{I2OS}(i, 4))$.
- c. Schritt 3: Falls 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: Falls i größer gleich $2^{32} = '1\ 0000\ 0000'$ ist, dann breche diesen Algorithmus mit der Fehlermeldung „error“ ab.
- f. Schritt 6: Falls das Achtfache von $\text{OctetLength}(n)$ kleiner als L_N ist, dann setze diesen Algorithmus mit Schritt 2 fort.
- g. Schritt 7: Setze $N = \text{Extract_MSBit}(n, L_N)$.

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:

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)

(N001.200) K_COS

Führe folgende Aktionen durch: 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:

Input:	x	beliebige positive oder negative reelle Zahl
Output:	n	kleinste ganze Zahl, die nicht kleiner ist als x
Errors:	–	Keine
Notation:		$n = \text{ceiling}(x)$

(N001.210) K_COS

Zu *x* MUSS die kleinste ganze Zahl *n* bestimmt werden, die nicht kleiner ist als *x*, das heißt: $\text{ceiling}(x) = \min \{n \in \mathbb{Z} \mid n \geq x\}$.

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:

Input:	x	beliebige positive oder negative reelle Zahl
Output:	n	größte ganze Zahl, die nicht größer ist als x
Errors:	–	Keine
Notation:		$n = \text{floor}(x)$

(N001.220) K_COS

Zu *x* MUSS die größte ganze Zahl *n* bestimmt werden, die nicht größer ist als *x*, das heißt: $\text{floor}(x) = \max \{n \in \mathbb{Z} \mid n \leq x\}$.

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

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. <i>Hinweis (15): Dieser Fehler ist für Smartcards nicht praxisrelevant. Um eine Nachricht mit einer Länge von 264 Bit innerhalb von 10 Jahren zu hashen, wäre der Hash-Algorithmus mit einer Geschwindigkeit von 6,8 GiB pro Sekunde auszuführen.</i>
Notation:		$H = \text{SHA_1}(M)$

(N001.290) K_COS

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

6.1.2 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 <i>Hinweis (16): 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.</i>
Notation:		$H = \text{SHA_256}(M)$

(N001.300) K_COS

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

6.1.3 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 <i>Hinweis (17): Dieser Fehler ist für Smartcards weniger relevant, als der unter Hinweis (16): beschriebene.</i>
Notation:		$H = \text{SHA_384}(M)$

(N001.310) K_COS

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

6.1.4 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 <i>Hinweis (18): Dieser Fehler ist für Smartcards weniger relevant, als der unter Hinweis (16): beschriebene.</i>
Notation:		$H = \text{SHA_512}(M)$

(N001.320) K_COS

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 3DES-Schlüssel, Option_DES

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 3DES-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 Funktionen in 6.7)

	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:		$(K_{enc}, T_1, K_{mac}, T_2) = \text{KeyDerivation_3TDES}(KD)$

(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.

- Schritt 1: $km = \text{BS2OS}(\text{MGF}(\text{OS2BS}(KD), 512, 1))$.
- Schritt 2: Teile km so auf K_{enc} , t_1 , K_{mac} und t_2 auf, dass gilt:
 - $\text{OctetLength}(K_{enc}) = \text{OctetLength}(K_{mac}) = 24$
 - $\text{OctetLength}(t_1) = \text{OctetLength}(t_2) = 8$
 - $km = K_{enc} \parallel t_1 \parallel K_{mac} \parallel t_2$
- Schritt 3: $T_1 = \text{OS2I}(t_1)$ und $T_2 = \text{OS2I}(t_2)$

6.2.2 Vereinbarung von AES-128-Schlüsseln

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 verwendet wird (siehe (N032.800) und (N034.100))
Errors:	–	keine
Notation:		$(K_{enc}, K_{mac}, T_2) = \text{KeyDerivation_AES128}(KD)$

(N001.500) K_COS

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

- $K_{enc} = \text{Extract_MSByte}(\text{SHA}_1(KD \parallel '00000001'), 16)$
- $K_{mac} = \text{Extract_MSByte}(\text{SHA}_1(KD \parallel '00000002'), 16)$
- $T_2 = 0$

6.2.3 Vereinbarung von AES-192-Schlüsseln

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 verwendet wird (siehe (N032.800) und (N034.100))
Errors:	–	keine
Notation:		$(K_{enc}, K_{mac}, T_2) = \text{KeyDerivation_AES192}(KD)$

(N001.510) K_COS

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

- $K_{enc} = \text{Extract_MSByte}(\text{SHA_256}(KD \parallel '00000001'), 24)$
- $K_{mac} = \text{Extract_MSByte}(\text{SHA_256}(KD \parallel '00000002'), 24)$
- $T_2 = 0$

6.2.4 Vereinbarung von AES-256-Schlüsseln

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 verwendet wird (siehe (N032.800) und (N034.100))
Errors:	–	keine
Notation:		$(K_{enc}, K_{mac}, T_2) = \text{KeyDerivation_AES256}(KD)$

(N001.520) K_COS

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

- $K_{enc} = \text{SHA_256}(KD \parallel '00000001')$
- $K_{mac} = \text{SHA_256}(KD \parallel '00000002')$
- $T_2 = 0$

6.2.5 Schlüsselableitung aus einer Card Access Number

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
	O/D	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:		$K = \text{KDF}(KD, n, O/D)$

(N001.530) K_COS

Option_kontaktlose_Schnittstelle, Option_PACE_PCD

Das COS MUSS K analog zu [BSI-TR-03110-3#A.2.3] wie folgt aus KD und O/D berechnen: Falls O/D aus der Menge

- {id-PACE-ECDH-GM-AES-CBC-CMAC-128, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128} ist, gilt:
 $K = \text{Extract_MSByte}(\text{SHA_1}(KD \parallel \text{I2OS}(n, 4)), 16)$
- {id-PACE-ECDH-GM-AES-CBC-CMAC-192, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192} ist, gilt:
 $K = \text{Extract_MSByte}(\text{SHA_256}(KD \parallel \text{I2OS}(n, 4)), 24)$
- {id-PACE-ECDH-GM-AES-CBC-CMAC-256, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256} ist, gilt:
 $K = \text{SHA_256}(KD \parallel \text{I2OS}(n, 4))$

6.3 Symmetrischer Basisalgorithmus für Vertraulichkeit

Ein Verschlüsselungsalgorithmus berechnet zu einem beliebigen Oktettstring (plaintext) mit Hilfe eines geheimen Schlüssels ein Chifftrat (ciphertext). Ein entsprechender Entschlüsselungsalgorithmus berechnet zu einem Chifftrat (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

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

Input:	P_j	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_j	Oktettstring, verschlüsselte Daten der Länge 8 Oktett
Errors:	–	Keine
Notation:		$C_j = \text{DES_ENC}(K, P_j)$

(N001.600) K_COS_G1, Option_DES

Das COS MUSS C_j mittels K gemäß [ANSI X3.92] aus P_j berechnen.

6.3.1.2 Verschlüsselung mittels 3TDES, Option_DES

Input:	P_j	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 K_a , K_b , K_c und es gilt: $K = K_a \parallel K_b \parallel K_c$
Output:	C_j	Oktettstring, verschlüsselte Daten der Länge 8 Oktett
Errors:	–	Keine
Notation:		$C_j = 3TDES_ENC(K, P_j)$

(N001.700) K_COS_G1, Option_DES

Das COS MUSS C_j mittels K wie folgt aus P_j berechnen:

$C_j = DES_ENC(K_c, DES_DEC(K_b, DES_ENC(K_a, P_j)))$.

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

Input:	C_j	Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Chiffre
	K	Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, der als Schlüssel verwendet wird
Output:	P_j	Oktettstring, entschlüsselte Daten der Länge 8 Oktett
Errors:	–	Keine
Notation:		$P_j = DES_DEC(K, C_j)$

(N001.800) K_COS_G1, Option_DES

Das COS MUSS P_j mittels K gemäß [ANSI X3.92] aus C_j berechnen.

6.3.2.2 Entschlüsselung mittels 3TDES, Option_DES

Input:	C_j	Beliebiger Oktettstring der Länge 8 Oktett = 64 Bit, Chiffre
	K	Beliebiger Oktettstring der Länge 24 Oktett = 192 Bit, der als Schlüssel verwendet wird. K setzt sich zusammen aus den drei Teilschlüsseln K_a , K_b , K_c und es gilt: $K = K_a \parallel K_b \parallel K_c$
Output:	P_j	Oktettstring, entschlüsselte Daten der Länge 8 Oktett
Errors:	–	Keine
Notation:		$P_j = 3TDES_DEC(K, C_j)$

(N001.900) K_COS_G1, Option_DES

Das COS MUSS P_j mittels K wie folgt aus C_j berechnen:

$$P_i = \text{DES_DEC}(K_a, \text{DES_ENC}(K_b, \text{DES_DEC}(K_c, C_i)))$$

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:

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:		$C_j = \text{AES_ENC}(K, P_j)$

(N002.000) K_COS

Das COS MUSS C_j mittels K so aus P_j berechnen, dass sich derselbe funktionale Zusammenhang $f_K: P_j \rightarrow C_j$ 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:

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:		$P_j = \text{AES_DEC}(K, C_j)$

(N002.010) K_COS

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

6.4 Asymmetrischer Basisalgorithmus RSA

Als asymmetrischer Basisalgorithmus wird RSA verwendet. Bezüglich des mathematischen Hintergrundes wird an dieser Stelle lediglich auf [PKCS#1] verwiesen.

(N002.100) K_COS

Das COS MUSS RSA-Schlüssel mit

- a. einer Modulslänge *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) K_COS

Das COS KANN weitere Modulslängen

- a. unterstützen oder
- b. ablehnen.

(N002.300) K_COS

Das COS KANN öffentliche Exponenten aus weiteren Intervallen

- a. unterstützen oder
- b. ablehnen.

(N002.400) K_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.

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

Tabelle 4: Liste der Schlüsselparameter eines RSA-Schlüssels

Parameter	Bedeutung
<i>n</i>	RSA Modulus
<i>e</i>	RSA öffentlicher Exponent
<i>d</i>	RSA privater Exponent

Hinweis (19): 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

Als asymmetrischer Basisalgorithmus werden elliptische Kurven verwendet. Bezüglich des mathematischen Hintergrundes wird an dieser Stelle lediglich auf [BSI-TR-03111] verwiesen. Die Domainparameter der im folgenden genannten Kurven finden sich auch im Anhang C.

(N002.500) K_COS

Das COS MUSS die folgenden elliptischen Kurven unterstützen:

- a. 256 Bit gemäß
 1. [RFC5639#3.4, brainpoolP256r1] (siehe C.3) und
 2. [ANSI X9.62#L6.4.3, ansix9p256r1] (siehe C.1).
- b. 384 Bit gemäß
 1. [RFC5639#3.6, brainpoolP384r1] (siehe C.4) und
 2. [ANSI X9.62#L6.5.2, ansix9p384r1] (siehe C.2).
- c. 512 Bit gemäß [RFC5639#3.7, brainpoolP512r1] (siehe C.5).

(N002.609) K_COS

Das COS KANN weitere elliptische Kurven

- a. unterstützen oder
- b. ablehnen.

6.6 Datenauthentisierung

Im Rahmen einer Datenauthentisierung werden beliebigen Daten 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

Input:	M	Beliebiger Oktettstring beliebiger Länge für den ein MAC berechnet wird
	K	Beliebiger Oktettstring, der als Schlüssel verwendet wird. Die Länge von K beträgt 24 Oktett. K setzt sich zusammen aus den drei Teilschlüsseln K_a , K_b , K_c und es gilt: $K = K_a \parallel K_b \parallel K_c$
Output:	T	Oktettstring, der zur Prüfung der Integrität und Authentizität von M verwendbar ist
Errors:	–	Keine
Notation:		$T = \text{CALCULATE_Retail_MAC}(K, M)$

(N002.700) K_COS_G1, Option_DES

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

- Schritt 1: Berechne $X = \text{PaddingIso}(M, 8)$
- Schritt 2: Teile X auf in Blöcke mit jeweils 8 Oktett $X = X_1 \parallel X_2 \parallel \dots \parallel X_n$.
- Schritt 3: Setze $Y_0 = '0000\ 0000\ 0000\ 0000'$.
- Schritt 4: Berechne $Y_i = \text{DES_ENC}(K, Y_{i-1} \text{ XOR } X_i)$ für $i = 1, \dots, n-1$.
- Schritt 5: Berechne $T = \text{3TDES_ENC}(K, Y_{n-1} \text{ XOR } X_n)$

Hinweis (20): 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

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:		$T = \text{CalculateCMAC_NoPadding}(K, M)$

(N002.810) K_COS

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

- Definitionen: CIPH = Block Cipher Algorithmus (siehe (N002.000))
 b = Blocklänge von CIPH in Bit hier gleich 128 Bit
- Schritt 1: Generiere Unterschlüssel $K1$ und $K2$ gemäß [CMAC#6.1].
- Schritt 2: Falls M_{len} gleich null ist, dann setze $n = 1$,
sonst $r = M_{len} \bmod b$ und
falls $r = 0$, dann setze $n = M_{len} / b$
sonst setze $n = (M_{len} - r) / b + 1$.
- Schritt 3: Teile M wie folgt in Blöcke auf:
 $M = M_1 \parallel M_2 \parallel \dots \parallel M_{n-1} \parallel M_n^*$
Die Blöcke M_1 bis M_{n-1} besitzen die Länge b . Der Block M_n^* besitzt eine Länge kleiner gleich b .
- Schritt 4: Falls $r = 0$ ist, dann setze $M_n = K1 \text{ XOR } M_n^*$,
sonst setze $M_n = K2 \text{ XOR } \text{PaddingIso}(M_n^*, b/8)$.
- Schritt 5: $C_0 = \text{I2OS}(0, b/8)$
- Schritt 6: $C_i = \text{AES_ENC}(K, C_{i-1} \text{ XOR } M_i)$.
- Schritt 7: $T = \text{Extract_MSByte}(C_n, 8)$.

6.6.1.3 Generierung CMAC mit internem ISO-Padding vor der CMAC-Berechnung

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:		$T = \text{CalculateCMAC_IsoPadding}(K, M)$

(N002.820) K_COS

Das COS MUSS T mittels K gemäß [CMAC#6.2] aus M wie folgt berechnen.

$T = \text{CalculateCMAC_NoPadding}(K, \text{PaddingIso}(M, 16))$.

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

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 24 Oktett. K setzt sich zusammen aus den drei Teilschlüsseln K_a , K_b , K_c und es gilt: $K = K_a \parallel K_b \parallel K_c$
Output:	out	Ergebnis der MAC-Prüfung, entweder VALID oder INVALID
Errors:	–	Keine
Notation:		$out = \text{VERIFY_Retail_MAC}(K, T', M)$

(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:

- Schritt 1: $T = \text{CALCULATE_Retail_MAC}(K, M)$.
- Schritt 2: Falls 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

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:	<i>out</i>	Ergebnis der MAC-Prüfung, entweder VALID oder INVALID
Errors:	–	Keine
Notation:		$out = \text{VerifyCMAC_NoPadding}(K, T', M)$

(N003.010) K_COS

Das COS MUSS die Integrität der Daten M mittels T' und K gemäß [CMAC#6.3] prüfen. Dabei werden folgende Schritte durchgeführt:

- Schritt 1: $T = \text{CalculateCMAC_NoPadding}(K, M)$.
- Schritt 2: Falls 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

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:	<i>out</i>	Ergebnis der MAC-Prüfung, entweder VALID oder INVALID
Errors:	–	Keine
Notation:		$out = \text{VerifyCMAC_IsoPadding}(K, T', M)$

(N003.020) K_COS

Das COS MUSS die Integrität der Daten M mittels T' und K gemäß [CMAC#6.3] prüfen. Dabei werden folgende Schritte durchgeführt:

- Schritt 1: $T = \text{CalculateCMAC_IsoPadding}(K, M)$.
- Schritt 2: Falls 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

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:

Input:	PrK	Privater RSA-Schlüssel gemäß 8.2.3
	M	Beliebiger Oktettstring, der die zu signierende Nachricht repräsentiert

Output:	<i>sig</i>	Oktettstring, welcher die Signatur repräsentiert
	M_2	Oktettstring, welcher den „non recoverable part“ der Nachricht M repräsentiert
Errors:	–	–
Notation:		$(sig, M_2) = \text{RSA_ISO9796_2_DS1_SIGN}(PrK, M)$

(N003.100) K_COS_G1, Option_RSA_CVC

Gemäß [ISO/IEC 9796-2#7, 8] MUSS das COS folgende Aktionen 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 werden MUSS und als Hash-Funktion SHA-256 (siehe (N001.300)) verwendet werden MUSS.
- 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: $J = \text{OS2I}(F)$
 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: $sig = \text{I2OS}(c, \text{OctetLength}(n))$

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:

Input:	<i>digestInfo</i>	Beliebiger Oktettstring, der als Digest Info verwendet wird (siehe [PKCS#1#A.2.4])
	<i>PrK</i>	Privater RSA-Schlüssel gemäß 8.2.3
Output:	<i>S</i>	Oktettstring, welcher die Signatur repräsentiert
Errors:	DigestInfoTooLong	Der Inputparameter <i>digestInfo</i> enthält zu viele Oktette
Notation:		$S = \text{RSASSA_PKCS1_V1_5_SIGN}(PrK, digestInfo)$

(N003.200) K_COS_G1

Gemäß [PKCS#1#8.2.1, 9.2] MUSS das COS folgende Aktion durchführen, wobei folgende Definitionen gelten: $n = PrK.n$, $d = PrK.d$.

- a. Schritt 0: Falls $\text{OctetLength}(digestInfo) > 0,4 \text{ OctetLength}(n)$ ist, dann breche diesen Algorithmus mit der Fehlermeldung DigestInfoTooLong ab.
- b. Schritt 1: Setze $EM \leftarrow '00' \parallel digestInfo$.
- c. Schritt 2: Setze $EM \leftarrow 'FF' \parallel EM$.

- d. Schritt 3: Falls $\text{OctetLength}(EM)$ kleiner als $\text{OctetLength}(n) - 2$ ist, dann fahre mit Schritt 2 fort.
- e. Schritt 4: Setze $EM \leftarrow '01' \parallel EM$.
- f. Schritt 5: Setze $m \leftarrow \text{OS2I}(EM)$.
- g. Schritt 6: Setze $s \leftarrow m^d \bmod n$.
- h. Schritt 7: Setze $S \leftarrow \text{I2OS}(s, \text{OctetLength}(n))$.

6.6.3.1.3 RSA, SSA, PSS

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

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 8.2.3
Output:	sig	Oktettstring, welcher die Signatur repräsentiert
Errors:	–	Keine
Notation:		$sig = \text{RSA_PSS_SIGN}(PrK, M_1, h(M_2))$

(N003.300) K_COS

Gemäß [ISO/IEC 9796-2#7, 9] MUSS das COS folgende Aktionen 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 werden MUSS und als Hash-Funktion SHA-256 verwendet werden MUSS. Im einzelnen:
 - 1. Setze $C = \text{I2OS}(\text{BitLength}(\text{OS2BS}(M_1)), 8)$
 - 2. Setze $S = \text{RAND}(32)$
 - 3. Berechne $H = \text{SHA_256}(C \parallel M_1 \parallel h(M_2) \parallel S)$
 - 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: $J = \text{OS2I}(F)$
 - 2. Schritt 2.2: $a = J^d \bmod n$
 - 3. Schritt 2.3: $sig = \text{I2OS}(a, \text{OctetLength}(n))$

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:

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 8.2.3
Output:	sig	Oktettstring, welcher die Signatur repräsentiert
Errors:	–	Keine
Notation:		$sig = \text{RSA_ISO9796_2_DS2_SIGN}(PrK, M_1, h(M_2))$

(N003.400) K_COS

Gemäß [ISO/IEC 9796-2#7, 9] MUSS das COS folgende Aktionen durchführen, wobei folgende Definition gilt: $n = PrK.n$

- Schritt 1: $a = \text{OS2I}(\text{RSA_PSS_SIGN}(PrK, M_1, h(M_2)))$.
- Schritt 2: $b = n - a$
- Schritt 3: $c = \min\{ a, b \}$
- Schritt 4: $sig = \text{I2OS}(c, \text{OctetLength}(n))$

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:

Input:	$mHash$	Beliebiger Oktettstring, der den Hash-Wert über die zu signierenden Nachricht M repräsentiert
	PrK	Privater RSA-Schlüssel gemäß 8.2.3
Output:	S	Oktettstring, welcher die Signatur repräsentiert
Errors:	–	Keine
Notation:		$S = \text{RSASSA_PSS_SIGN}(PrK, mHash)$

(N003.500) K_COS

Gemäß [PKCS#1#8.1.1, 9.1.1] MUSS das COS folgende Aktion durchführen:

- Schritt 1: $M_1 = ''$, (Anmerkung: M_1 ist ein leerer Oktettstring)
- Schritt 2: $S = \text{RSA_PSS_SIGN}(PrK, M_1, mHash)$.

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:

Input:	H	Oktettstring mit beliebigem Inhalt, der einen Hash-Wert repräsentiert
	PrK	Privater ELC-Schlüssel gemäß 8.2.3
Output:	R	Oktettstring, erster Teil der ECDSA-Signatur
	S	Oktettstring, zweiter Teil der ECDSA-Signatur
Errors:	–	Keine
Notation:		$(R, S) = \text{ELC_SIG}(PrK, H)$

(N003.600) K_COS

Gemäß [BSI-TR-03111#4.2.1.1] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten:

$$\begin{aligned} d_A &= PrK.d, & G &= PrK.domainParameter.G, \\ n &= PrK.domainParameter.n, & \tau &= PrK.domainParameter.\tau \end{aligned}$$

- a. Schritt 0: $H_\tau = \text{BS2OS}(\text{Extract_MSBit}(H, \tau))$.
- b. Schritt 1: $k = \text{RNG}(\{1, 2, \dots, n-1\})$, 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: $r = \text{OS2I}(\text{FE2OS}(x_Q)) \bmod n$.
Falls r gleich null ist, dann gehe zu Schritt 1.
- e. Schritt 4: $k_{inv} = k^{-1} \bmod n$.
- f. Schritt 5: $s = k_{inv}(r d_A + \text{OS2I}(H_\tau)) \bmod n$.
Falls s gleich null ist, dann gehe zu Schritt 1.
- g. Schritt 6: $R = \text{I2OS}(r, \text{ceiling}(\tau / 8))$.
 $S = \text{I2OS}(s, \text{ceiling}(\tau / 8))$.

Hinweis (21): 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_τ 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 nicht unmittelbar sichtbar. 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

Input:	PuK	Öffentlicher RSA-Schlüssel gemäß 8.2.4
--------	-----	--

	sig	Beliebiger Oktettstring, der eine Signatur repräsentiert
	M ₂	Beliebiger Oktettstring, der den „non recoverable part“ der Nachricht M repräsentiert. M ₂ KANN leer sein
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)

(N003.700) K_COS_G1, Option_RSA_CVC

Gemäß [ISO/IEC 9796-2#7.2] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten: $n = \text{PuK}.n$, $e = \text{PuK}.e$.

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

- Schritt 1.1: Falls die Bitlänge von *sig* ungleich der Bitlänge von *n* ist, dann gebe *False* zurück und breche diesen Algorithmus ab.
- Schritt 1.2: Falls das höchstwertige Bit von *sig* den Wert 1 hat, dann gebe *False* zurück und breche diesen Algorithmus ab.
- Schritt 1.3: $s = \text{OS2I}(sig)$.
- Schritt 1.4: $J^* = s^e \bmod n$.
- Schritt 1.5: Falls J^* gerade ist, dann setze $I^* = J^*$,
sonst setze $I^* = n - J^*$.
- Schritt 1.6: Falls $I^* \bmod 256$ ungleich 'BC' ist, dann gebe *False* zurück und breche diesen Algorithmus ab.
- Schritt 1.7: $F^* = \text{I2OS}(I^*, \text{OctetLength}(n))$.
- Schritt 1.8: Falls das höchstwertige Bit von F^* den Wert 1 hat, dann gebe *False* zurück und breche diesen Algorithmus ab.

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

- Schritt 2.1: Falls das zweithöchste Bit von F^* den Wert 0 hat, dann gebe *False* zurück und breche diesen Algorithmus ab.
- Schritt 2.2: Berechne H^* und M_1^* gemäß [ISO/IEC 9796-2#8.4].
- Schritt 2.3: Berechne $M = M_1 \parallel M_2$.
- Schritt 2.4: Berechne $H = \text{SHA_256}(M)$.
- Schritt 2.5: Falls H identisch ist zu H^* , dann gebe *True* und M zurück, sonst gebe *False* ohne M zurück.

Hinweis (22): ACHTUNG: In (N003.700)b.1 ist F^ ein Oktettstring. Aus diesem F^* geht durch Abschneiden des „most significant bit“ ein Bitstring hervor, der F^* aus [ISO/IEC 9796-2#8.4] entspricht. Deshalb wird in (N003.700)b.1 das zweithöchste Bit geprüft und in [ISO/IEC 9796-2#8.4] das „leftmost“ Bit.*

6.6.4.2 Signaturprüfung mittels elliptischer Kurven

Input:	PuK	Öffentlicher ELC-Schlüssel gemäß 8.2.4
	H	Beliebiger Oktettstring, der einen Hash-Wert repräsentiert

	<i>R</i>	Oktettstring, erster Teil der ECDSA-Signatur
	<i>S</i>	Oktettstring, zweiter Teil der ECDSA-Signatur
Output:	<i>out</i>	Boolean, True, falls die Signatur gültig ist, andernfalls False
Errors:	–	Keine
Notation:		$out = ELC_VER_SIG(PuK, R, S, H)$

(N003.800) K_COS

Gemäß [BSI-TR-03111#4.2.1.2] MUSS das COS folgende Aktionen durchführen, wobei folgende Definitionen gelten:

$$P_a = PuK.P, \quad G = PuK.domainParameter.G, \\ n = PuK.domainParameter.n, \quad \tau = PuK.domainParameter.\tau$$

- a. Schritt 0: $H_\tau = BS2OS(Extract_MSBit(H, \tau))$, siehe Hinweis (21):
 $r = OS2I(R)$.
 $s = OS2I(S)$.
- b. Schritt 1: Überprüfe, ob r und s Element der Menge $\{1, 2, \dots, n-1\}$ sind. Falls nicht, gebe *False* zurück und breche diesen Algorithmus ab.
- c. Schritt 2: $s_{inv} = s^{-1} \bmod n$.
- d. Schritt 3: $u_1 = s_{inv} OS2I(H_\tau) \bmod n$.
 $u_2 = s_{inv} r \bmod n$.
- e. Schritt 4: $Q = [u_1] G + [u_2] P_a$ mit $Q = (x_Q, y_Q)$.
- f. Schritt 5: $v = OS2I(FE2OS(x_Q)) \bmod n$.
- g. Schritt 6: Gebe *True* zurück, falls v gleich r ist, andernfalls *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 Chifftrat *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

Input:	<i>P</i>	Oktettstring, Klartext (plaintext), beliebiger Oktettstring beliebiger Länge, der verschlüsselt wird
	<i>K</i>	Beliebiger Oktettstring der Länge 24 Oktett, der als Schlüssel verwendet wird
	<i>T₁</i>	Beliebige nicht-negative Zahl, die als Startwert verwendet wird
Output:	<i>C</i>	Oktettstring, Chifftrat (ciphertext)
Errors:	<i>lengthError</i>	Die Länge von <i>P</i> ist kein ganzzahliges Vielfaches der Blocklänge
Notation:		$C = 3TDES_CBC_ENC(K, T_1, P)$

(N003.900) K_COS_G1, Option_DES

Das COS MUSS C mittels K und T_1 wie folgt aus P berechnen. Dabei sind folgende Aktionen durchzuführen:

- Schritt 1: Falls $\text{OctetLength}(P) \bmod 8$ ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
- Schritt 2: Teile P auf in Blöcke mit jeweils 64 Bit $P = P_1 \parallel P_2 \parallel \dots \parallel P_n$.
- Schritt 3: Setze $C_0 = \text{I2OS}(T_1, 8)$
- Schritt 4: Berechne $C_i = \text{3DES_ENC}(K, C_{i-1} \text{ XOR } P_i)$ für $i = 1, \dots, n$.
- Schritt 5: Berechne $C = C_1 \parallel C_2 \parallel \dots \parallel C_{n-1} \parallel C_n$.

6.7.1.2 Verschlüsselung AES

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 verwendet wird
Output:	C	Oktettstring, Chiffre (ciphertext), das dieselbe Länge wie P besitzt
Errors:	<i>lengthError</i>	Die Länge von P ist kein ganzzahliges Vielfaches der Blocklänge
Notation:		$C = \text{AES_CBC_ENC}(K, T_1, P)$

(N004.000) K_COS

Das COS MUSS C mittels K und T_1 gemäß [NIST sp800-38a#6.2] aus P berechnen. Dabei sind folgende Aktionen durchzuführen:

- Schritt 1: Falls $\text{OctetLength}(P) \bmod 16$ ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
- Schritt 2: Teile P auf in Blöcke mit jeweils 128 Bit $P = P_1 \parallel P_2 \parallel \dots \parallel P_n$.
- Schritt 3: Setze $C_0 = \text{I2OS}(T_1, 16)$
- Schritt 4: Berechne $C_i = \text{AES_ENC}(K, C_{i-1} \text{ XOR } P_i)$ für $i = 1, \dots, n$.
- Schritt 5: Berechne $C = C_1 \parallel C_2 \parallel \dots \parallel C_{n-1} \parallel C_n$.

6.7.2 Symmetrische Entschlüsselung

Die symmetrische Entschlüsselung überführt ein Chiffre *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 Verschlüsselung als Funktion wie folgt verwendet:

6.7.2.1 Entschlüsselung 3DES, Option_DES

Input:	C	Beliebiger Oktettstring, Chiffre (ciphertext), der entschlüsselt wird, die Länge ist ein ganzzahliges Vielfaches der Blocklänge
--------	-----	---

	K	Beliebiger Oktettstring der Länge 192 Bit, der als Schlüssel verwendet wird
	T_1	Beliebige nicht-negative Zahl, die als Startwert verwendet wird
Output:	P	Oktettstring, Klartext (plaintext)
Errors:	<i>lengthError</i>	Die Länge von C ist kein ganzzahliges Vielfaches der Blocklänge
Notation:		$P = 3TDES_CBC_DEC(K, T_1, C)$

(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:

- Schritt 0: Falls $\text{OctetLength}(P) \bmod 8$ ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
- Schritt 1: Teile C auf in Blöcke mit jeweils 64 Bit $C = C_1 \parallel C_2 \parallel \dots \parallel C_n$.
- Schritt 2: Setze $C_0 = \text{I2OS}(T_1, 8)$
- Schritt 3: Berechne $P_i = 3TDES_DEC(K, C_i) \text{ XOR } C_{i-1}$ für $i = 1, \dots, n$.
- Schritt 4: Berechne $P = P_1 \parallel P_2 \parallel \dots \parallel P_{n-1} \parallel P_n$.

6.7.2.2 Entschlüsselung AES

Input:	C	Beliebiger Oktettstring, Chiffre (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 verwendet wird
Output:	P	Oktettstring, Klartext (plaintext), der dieselbe Länge wie C besitzt
Errors:	<i>lengthError</i>	Die Länge von C ist kein ganzzahliges Vielfaches der Blocklänge
Notation:		$P = \text{AES_CBC_DEC}(K, T_1, C)$

(N004.200) K_COS

Das COS MUSS P mittels K und T_1 gemäß [NIST sp800-38a#6.2] aus C berechnen. Dabei sind folgende Aktionen durchzuführen:

- Schritt 0: Falls $\text{OctetLength}(P) \bmod 16$ ungleich 0 ist, dann gebe den Fehler *lengthError* zurück und breche diesen Algorithmus ab.
- Schritt 1: Teile C auf in Blöcke mit jeweils 128 Bit $C = C_1 \parallel C_2 \parallel \dots \parallel C_n$.
- Schritt 2: Setze $C_0 = \text{I2OS}(T_1, 16)$
- Schritt 3: Berechne $P_i = \text{AES_DEC}(K, C_i) \text{ XOR } C_{i-1}$ für $i = 1, \dots, n$.
- Schritt 4: Berechne $P = P_1 \parallel P_2 \parallel \dots \parallel P_{n-1} \parallel P_n$.

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 Chiffre 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

Input:	PuK	Öffentlicher RSA-Schlüssel gemäß 8.2.4
	M	Beliebiger Oktettstring, zu verschlüsselnde Nachricht
Output:	C	Oktettstring, Chiffre zur Nachricht M
Errors:	<i>ERROR</i>	„message too long“, falls M ist zu lang
Notation:		$C = \text{RSAES_PKCS1_V1_5_ENCRYPT}(PuK, M)$

(N004.300) K_COS

Das COS MUSS C mittels PuK und M gemäß [PKCS#1#7.2.1] berechnen. Es gelten folgende Definitionen: $n = PuK.n$, $e = PuK.e$.

- Schritt 0: Setze $Plen = \text{OctetLength}(n) - \text{OctetLength}(M) - 3$.
- Falls $Plen$ kleiner als acht ist, dann breche diesen Algorithmus mit der Fehlermeldung „ERROR“ ab.
- Schritt 1: Setze $EM = '00' \parallel M$.
- Schritt 2: Setze $PS = \text{RAND}(1)$.
- Schritt 3: Falls PS gleich '00' ist, fahre mit Schritt 2 fort.
- Schritt 4: Setze $EM = PS \parallel EM$.
- Schritt 5: Falls $\text{OctetLength}(EM)$ kleiner als $\text{OctetLength}(n) - 2$ ist, dann fahre mit Schritt 2 fort.
- Schritt 6: Setze $EM = '02' \parallel EM$.
- Schritt 7: Setze $m = \text{OS2I}(EM)$.
- Schritt 8: Setze $c = m^e \bmod n$.
- Schritt 9: Setze $C = \text{I2OS}(c, \text{OctetLength}(n))$.

6.8.1.2 RSA, OAEP, Verschlüsselung

Input:	PuK	Öffentlicher RSA-Schlüssel gemäß 8.2.4
	M	Beliebiger Oktettstring, zu verschlüsselnde Nachricht
Output:	C	Oktettstring, Chiffre zur Nachricht M
Errors:	<i>ERROR</i>	„message too long“, falls M ist zu lang
Notation:		$C = \text{RSAES_OAEP_ENCRYPT}(PuK, M)$

(N004.400) K_COS

Das COS MUSS C mittels PuK und M gemäß [PKCS#1#7.1.1] berechnen. Es gelten folgende Definitionen: $n = PuK.n$, $e = PuK.e$.

- a. Schritt 1: Falls $\text{OctetLength}(M)$ größer als $\text{OctetLength}(n) - 66$ ist, dann breche diesen Algorithmus mit dem Fehler „ERROR“ ab.
- b. Schritt 2: Setze $L = ''$, (Anmerkung: leerer Oktettstring).
- c. Schritt 3: Setze $IHash = \text{SHA}_{256}(L)$.
- d. Schritt 4: Setze $Plen = \text{OctetLength}(n) - \text{OctetLength}(M) - 66$.
- e. Schritt 5: Setze $PS = \text{I2OS}(0, Plen)$.
- f. Schritt 6: Setze $DB = IHash \parallel PS \parallel '01' \parallel M$.
- g. Schritt 7: Setze $seed = \text{RAND}(32)$.
- h. Schritt 8: Setze $dbMask = \text{BS2OS}(\text{MGF}(\text{OS2BS}(seed), 8 \cdot (\text{OctetLength}(n) - 33), 0))$.
- i. Schritt 9: Setze $maskedDB = DB \text{ XOR } dbMask$.
- j. Schritt 10: Setze $seedMask = \text{BS2O2}(\text{MGF}(\text{OS2BS}(maskedDB), 8 \cdot \text{OctetLength}(IHash), 0))$.
- k. Schritt 11: Setze $maskedSeed = seed \text{ XOR } seedMask$.
- l. Schritt 12: Setze $EM = '00' \parallel maskedSeed \parallel maskedDB$.
- m. Schritt 13: Setze $m = \text{OS2I}(EM)$.
- n. Schritt 14: Setze $c = m^e \bmod n$.
- o. Schritt 15: Setze $C = \text{I2OS}(c, \text{OctetLength}(n))$.

6.8.1.3 Elliptic Curve Key Agreement

Diese Funktionalität ist an der physikalischen Schnittstelle nicht direkt sichtbar, wird aber im Rahmen der asymmetrischen Ver- und Entschlüsselung mittels elliptischer Kurven verwendet.

Input:	d	natürliche Zahl, die dem privaten Schlüssel PrK entspricht
	P	Punkt auf derselben elliptischen Kurve, wie PrK
	dP	Domainparameter gemäß (N008.600)
Output:	K_{AB}	„gemeinsames Geheimnis“
Errors:	ERROR	Falls das Ergebnis in Schritt 3 der unendlich ferne Punkt ist
Notation:		$K_{AB} = \text{ECKA}(d, P, dP)$

(N004.490) K_COS

Das COS MUSS K_{AB} mittels d , P und dP berechnen, wobei die Schritte 1 bis 4 [BSI-TR-03111#4.3.1] entsprechen. Es gelten folgende Definitionen:

$$h = dP.h, \quad n = dP.n, \quad L = dP.L$$

- a. Schritt 1: $l = h^{-1} \bmod n$.
- b. Schritt 2: $Q = [h] P$.
- c. Schritt 3: $S = [d l \bmod n] Q$. mit $S = (x_S, y_S)$.
Falls S gleich dem unendlich fernen Punkt O der Kurve ist, dann gebe den Fehler „ERROR“ zurück und beende diesen Algorithmus.
- d. Schritt 4: $K_{AB} = \text{I2OS}(x_S, L)$.

6.8.1.4 ELC Verschlüsselung

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, ephemere Punkt PE_A des Senders
	C	Oktettstring, Chiffre der Nachricht M
	T	Oktettstring, MAC über das Chiffre C
Errors:	ERROR	<ul style="list-style-type: none"> Falls in Schritt 0 ein Fehler auftritt Falls in ECKA ein Fehler auftritt
Notation:		$(PO_A, C, T) = \text{ELC_ENC}(M, PO_B, dP)$

(N004.500) K_COS

Das COS MUSS PO_A , C und T mittels M und PO_B berechnen, wobei die Schritt 3 [BSI-TR-03111#4.3.1] entspricht und folgende Definitionen gelten:

$$\begin{aligned} n &= dP.n, & h &= dP.h, \\ G &= dP.G, & L &= dP.L \end{aligned}$$

- a. Schritt 0: $P_B = \text{OS2P}(PO_B, dP)$.
Falls diese Funktion mit einem Fehler terminiert, dann gebe „ERROR“ zurück und beende diesen Algorithmus.
- b. Schritt 1: $d = \text{RNG}(\{1, 2, \dots, n-1\})$, d.h. zufällig erzeugte ganze Zahl aus dem Intervall $[1, n-1]$.
- c. Schritt 2: $PE_A = [d] G$.
- d. Schritt 3: $K_{AB} = \text{ECKA}(d, P_B, dP)$.
Falls die Funktion ECKA einen Fehler meldet, dann gebe den Fehler „ERROR“ zurück und beende diesen Algorithmus.
- e. Schritt 4: Berechne abgeleitete Schlüssel gemäß (N001.520)
 1. $(K_{enc}, K_{mac}, T_2) = \text{KeyDerivation_AES256}(K_{AB})$.
 2. Setze: $T_1 = \text{OS2I}(\text{AES_ENC}(K_{enc}, \text{I2OS}(T_2, 16)))$
- f. Schritt 5: $PO_A = \text{P2OS}(PE_A, L)$.
- g. Schritt 6: $C = \text{AES_CBC_ENC}(K_{enc}, T_1, \text{PaddingIso}(M, 16))$.

h. Schritt 7: $T = \text{CalculateCMAC_IsoPadding}(K_{mac}, C)$.

6.8.2 Asymmetrische Entschlüsselung

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

6.8.2.1 RSA, ES, PKCS1 V1.5, Decrypt

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

Input:	PrK	ein privater RSA-Schlüssel gemäß 8.2.3
	C	beliebiger Oktettstring, Chifftrat der Nachricht M
Output:	M	Oktettstring, Klartextnachricht zum Chifftrat
Errors:	ERROR	„decryption error“, falls einer der folgenden Fälle eintritt: C ist numerisch größer gleich dem Modulus EM fängt nicht mit '0002' an In EM gibt es kein '00' Oktett, welches PS von M trennt PS hat weniger als acht Oktett
Notation:		$M = \text{RSAES_PKCS1_V1_5_DECRYPT}(PrK, C)$

(N004.600) K_COS

Das COS MUSS M mittels PrK und C gemäß [PKCS#1#7.2.2] berechnen. Es gelten folgende Definitionen: $n = PrK.n$, $d = PrK.d$.

- Schritt 1: Falls $\text{OctetLength}(C)$ ungleich $\text{OctetLength}(n)$ ist, breche diesen Algorithmus mit „decryption error“ ab.
- Schritt 2: Setze $c = \text{OS2I}(C)$.
- Schritt 3: Falls c größer oder gleich n ist, breche diesen Algorithmus mit „decryption error“ ab.
- Schritt 4: Setze $EM = \text{I2OS}(c^d \bmod n, \text{OctetLength}(n))$.
- Falls das erste Oktett in EM ungleich '00' ist, oder das zweite Oktett in EM ungleich '02' ist, breche diesen Algorithmus mit „decryption error“ ab.
- Schritt 5: Teile EM wie folgt auf: $EM = '00\ 02 \parallel PS \parallel 00 \parallel M'$. Dabei MUSS jedes Oktett in PS ungleich '00' sein.
- Schritt 6: Breche diesen Algorithmus mit „decryption error“ ab, falls
 - PS kürzer als 8 Oktett ist, oder
 - kein Oktett mit dem Wert '00' existiert, welches PS von M trennt.
- Schritt 7: Gebe M zurück.

Hinweis (23): Die Prüfung in Schritt 1 ist überflüssig, da wegen (N089.000) der in Schritt 1 geprüfte Fall im Rahmen dieser Spezifikation nicht auftritt. Zudem wird in Schritt 3 strenger geprüft.

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).

Input:	<i>PrK</i>	Privater RSA-Schlüssel gemäß 8.2.3
	<i>C</i>	Beliebiger Oktettstring, Chiffre der Nachricht <i>M</i>
Output:	<i>M</i>	Oktettstring, Klartextnachricht zum Chiffre
Errors:	ERROR	„decryption error“, falls einer der folgenden Fälle eintritt: <i>C</i> ist numerisch größer gleich dem Modulus von <i>PrK</i>
Notation:		$M = \text{RSAES_OAEP_DECRYPT}(PrK, C)$

(N004.700) K_COS

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 $c = \text{OS2I}(C)$.
- c. Schritt 3: Falls c größer gleich n ist, dann breche diesen Algorithmus mit dem Fehler „decryption error“ ab.
- d. Schritt 4: Setze $m = c^d \bmod n$.
- e. Schritt 5: Setze $EM = \text{I2OS}(m, \text{OctetLength}(n))$.
- f. Schritt 6: Setze $IHash = \text{SHA}_{256}(L)$.
- g. Schritt 7: Teile EM wie folgt auf:
 1. Schritt 7.1: $EM = Y \parallel \text{maskedSeed} \parallel \text{maskedDB}$.
 2. Schritt 7.2: $1 = \text{OctetLength}(Y)$.
 3. Schritt 7.3: $32 = \text{OctetLength}(\text{maskedSeed})$.
- h. Schritt 8: Setze $\text{seedMask} = \text{BS2OS}(\text{MGF}(\text{OS2BS}(\text{maskedDB}), 8 \cdot \text{OctetLength}(IHash), 0))$.
- i. Schritt 9: Setze $\text{seed} = \text{maskedSeed} \text{ XOR } \text{seedMask}$.
- j. Schritt 10: Setze $\text{dbMask} = \text{BS2OS}(\text{MGF}(\text{OS2BS}(\text{seed}), 8 \cdot \text{OctetLength}(\text{maskedDB}), 0))$.
- k. Schritt 11: Setze $DB = \text{maskedDB} \text{ XOR } \text{dbMask}$.
- l. Schritt 12: Teile DB wie folgt auf:
 1. Schritt 12.1: $DB = IHash' \parallel PS \parallel '01' \parallel M$.
 2. Schritt 12.2: $32 = \text{OctetLength}(IHash')$.

3. Schritt 12.3: Der möglicherweise leere Oktettstring PS darf nur Oktette mit dem Wert '00' enthalten.
- m. Schritt 13: Breche diesen Algorithmus mit „decryption error“ ab, wenn
 1. $IHash'$ ungleich $IHash$ 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 Asymmetrische Entschlüsselung mittels ELC

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

Input:	PrK	Privater ELC-Schlüssel gemäß 8.2.3
	PO	Oktettstring, ephemerer Punkt PE_A des Senders
	C	Oktettstring, Chiffre der Nachricht M
	T'	Oktettstring, MAC über das Chiffre C
Output:	M	Oktettstring, Klartextnachricht zum Chiffre
Errors:	ERROR	Falls PO nicht im Format „uncompressed encoding“ vorliegt Falls PO einen Punkt bezeichnet, der nicht auf derselben Kurve liegt, wie PrK Falls die Funktion ECKA einen Fehler meldet Falls die MAC-Prüfung fehlschlägt
Notation:		$M = \text{ELC_DEC}(PO, PrK, C, T')$

(N004.800) K_COS

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:

$$\begin{aligned} d &= PrK.d, & h &= PrK.domainParameter.h, \\ n &= PrK.domainParameter.n, & L &= PrK.domainParameter.L \end{aligned}$$

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

7 CV-Zertifikat

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, Option_RSA_CVC

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 im 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 eindeutig 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).

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 eindeutig 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 eindeutig.

Im Rahmen von 7.1 werden nur folgende OID-Werte verwendet (siehe Tabelle 271) :
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:

(N004.900) K_externeWelt {K_Karte}, Option_RSA_CVC

Der CPI MUSS den Wert '21' haben.

(N005.000) K_externeWelt {K_Karte}, Option_RSA_CVC

Die OID MUSS den Wert sigS_ISO9796-2Withrsa_sha256 = '2B24 0304 0202 04' haben (siehe Tabelle 271).

(N005.100) K_externeWelt {K_Karte}, Option_RSA_CVC

Die Länge des Modulus MUSS 2048 bit betragen.

(N005.200) K_externeWelt {K_Karte}, Option_RSA_CVC
Der öffentliche Exponent MUSS vier Oktett lang sein.

(N005.300) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CHR MUSS acht Oktett lang sein.

(N005.400) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CAR MUSS acht Oktett lang sein.

(N005.500) K_externeWelt {K_Karte}, Option_RSA_CVC
Für die zu signierende Nachricht *M* gilt:
 $M = \text{CPI} \parallel \text{Modulus} \parallel \text{öffentlicherExponent} \parallel \text{OID} \parallel \text{CHR} \parallel \text{CAR}.$

Wenn (N002.100)a berücksichtigt wird und der Hash-Wert gemäß 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:

(N005.600) K_externeWelt {K_Karte}, Option_RSA_CVC
Der CPI MUSS den Wert '22' haben.

(N005.700) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CHA MUSS sieben Oktett lang sein.

(N005.800) K_externeWelt {K_Karte}, Option_RSA_CVC
Die OID MUSS den Wert `authS_ISO9796-2Withrsa_sha256_mutual = '2B24 0305 0204'` haben (siehe Tabelle 271).

(N005.900) K_externeWelt {K_Karte}, Option_RSA_CVC
Die Länge des Modulus MUSS 2048 bit betragen.

(N006.000) K_externeWelt {K_Karte}, Option_RSA_CVC
Der öffentliche Exponent MUSS vier Oktett lang sein.

(N006.100) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CHR MUSS zwölf Oktett lang sein.

(N006.200) K_externeWelt {K_Karte}, Option_RSA_CVC
Die CAR MUSS acht Oktett lang sein.

(N006.300) K_externeWelt {K_Karte}, Option_RSA_CVC
Für die zu signierende Nachricht *M* gilt:
 $M = \text{CPI} \parallel \text{Modulus} \parallel \text{öffentlicherExponent} \parallel \text{OID} \parallel \text{CHA} \parallel \text{CHR} \parallel \text{CAR}.$

Wenn (N002.100)a berücksichtigt wird und der Hash-Wert gemäß 7.1.2 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:

(N006.400) K_externeWelt {K_Karte}, Option_RSA_CVC
Schritt 1: Die Nachricht *M* MUSS gemäß 7.1.2 erzeugt werden.

(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:

$$(SIG.CA, M_2) = \text{RSA_ISO9796_2_DS1_SIGN}(PrK, M).$$

(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:

- Datenelement $SIG.CA$ als Wertfeld in einem Datenobjekt mit Tag = '5F37'.
- 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 5: 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 M2 ('003E' = 62 Oktett)

Tabelle 6: 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 M2 ('0048' = 72 Oktett)

Hinweis (24): Vorgaben für CV-Zertifikate für ELC-Schlüssel finden sich in [gemSpec_PKI].

7.2 CV-Zertifikate für ELC-Schlüssel (informativ)

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:

(N006.700) K_Anwendungsspezifikation {K_Karte}

Der Wert von *fileIdentifier* MUSS eine ganze Zahl im Intervall [‘1000’, ‘FEFF’] oder Element der Menge {‘011C’} sein.

(N006.800) K_Anwendungsspezifikation {K_Karte}

Ein Objekt, das nicht *root* (siehe 9.1) ist, DARF KEINEN *fileIdentifier* mit dem Wert ‘3F00’ besitzen.

(N006.900) K_Anwendungsspezifikation {K_Karte}

KEIN Objekt DARF einen *fileIdentifier* mit dem Wert ‘3FFF’ besitzen.

Hinweis (25): Die Forderung (N006.700) ist strenger als [ISO/IEC 7816-4]. Dies lässt Raum für herstellersistem spezifische 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:

(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#Abbildung 1] leiten sich folgende Regeln ab, die bei der Spezifikation einer Anwendung einzuhalten sind:

(N007.100) K_Anwendungsspezifikation {K_Karte}

- a. Der Wert von *lifeCycleStatus* MUSS ein Element der Menge {
„Operational state (active)“,
„Operational state (deactivated)“,
„Termination state“
} sein.
- b. Ein COS KANN weitere Werte für *lifeCycleStatus*
 - 1. unterstützen oder
 - 2. ablehnen.

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 mit Zugriffsregeln angebracht, wo je nach Interface (kontaktbehaftet/kontaktlos = erste Dimension) Lebenszyklus (aktiviert/deaktiviert = zweite Dimension) und Einsatzumgebung (SE-Identifizier = 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 11.2.2 oder 11.2.1 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 schlichte Listen verwendet.

(N007.170) K_Anwendungsspezifikation {K_Karte}

- a. Die Liste *interfaceDependentAccessRules* MUSS zwei Elemente umfassen.
- b. 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.
- c. 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.
- d. Ein COS KANN *interfaceDependentAccessRules* Listen mit mehr Elementen
 - 1. unterstützen oder
 - 2. ablehnen.
- e. Ein COS, welches die Option_kontaktlose_Schnittstelle nicht anbietet, KANN das zweite Listenelement
 - 1. beachten oder
 - 2. ignorieren.

(N007.200) K_Anwendungsspezifikation {K_Karte}

- a. Der Wert von *accessRuleList* MUSS eine Liste mit mindestens drei und maximal sechs Elementen sein.
- b. Ein COS KANN *accessRuleList* mit mehr Elementen
 - 1. unterstützen oder
 - 2. ablehnen.

(N007.300) K_Anwendungsspezifikation {K_Karte}

- a. Ein Listenelement MUSS für den Wert der Variable *lifeCycleStatus* = „Operational state (deactivated)“ (siehe 8.1.3) verwendet werden.
- b. Ein weiteres Listenelement MUSS für den Wert der Variable *lifeCycleStatus* = „Termination state“ (siehe 8.1.3) verwendet werden.

(N007.400) K_Anwendungsspezifikation {K_Karte}

Die übrigen Listenelemente MÜSSEN genau einem *selfIdentifier* Wert gemäß (N007.900) zugeordnet werden.

(N007.500) K_Anwendungsspezifikation {K_Karte}

Jedes Listenelement 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.

(N007.600) K_Anwendungsspezifikation {K_Karte}

- a. Ein *record* MUSS eine Rekordnummer *number* besitzen. Die Rekordnummer MUSS eine ganze Zahl im Intervall [1, 254] sein.
- b. Ein COS KANN Rekordnummern mit weiteren Werten
 - 1. unterstützen oder
 - 2. ablehnen.

(N007.700) K_Anwendungsspezifikation {K_Karte}

- a. Ein *record* MUSS einen Oktettstring *data* besitzen, dessen Anzahl enthaltener Oktette im Intervall [‘01’, ‘FF’] = [1, 255] liegen MUSS.
- b. Ein COS KANN weitere Längen für *record*
 - 1. unterstützen oder
 - 2. ablehnen.

(N007.800) K_Anwendungsspezifikation {K_Karte}

Ein *record* MUSS eine Liste mit Attributen vom Typ *lifeCycleStatus* (siehe 8.1.3) unterstützen. Diese Liste ist entweder

- a. leer (der *record* hat keinen *lifeCycleStatus* und befindet sich implizit im Zustand „Operational state (active)“) oder
- b. enthält genau ein Element (der *record* hat genau einen *lifeCycleStatus*). Dieses Element MUSS einen Wert aus der folgenden Menge besitzen:
{„Operational state (active)“, „Operational state (deactivated)“}.

Hinweis (26): Da es aus funktionaler Sicht keine Möglichkeit gibt den lifeCycleStatus eines record in Zustand „Termination state“ zu überführen, ist dieser Zustand für einen record irrelevant.

8.1.6 SE-Identifizier

Der Attributstyp *seldentifier* 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:

(N007.900) K_Anwendungsspezifikation {K_Karte}

- a. Der Wert von *seldentifier* MUSS eine ganze Zahl im Intervall [1, 4] sein.
- b. Das COS KANN weitere Werte für *seldentifier*
 1. akzeptieren oder
 2. ablehnen.

8.1.7 PIN

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

(N008.000) K_Anwendungsspezifikation {K_Karte}

- a. Ein *pin* MUSS eine Folge von Ziffern sein, die aus dem Wertebereich {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} stammen MÜSSEN. Die Anzahl der Ziffern MUSS im Intervall [4, 12] liegen.
- b. Das COS KANN weitere Zeichen oder weitere Längen für *pin*
 1. unterstützen oder
 2. ablehnen.

(N008.100) K_Anwendungsspezifikation {K_Karte}

Für die Umwandlung des Attributtyps *pin* nach Format-2-PIN-Block gilt:

- a. Der Format-2-PIN-Block ist ein Oktettstring mit acht Oktett = 16 Nibble.
- b. Das erste Nibble MUSS den Wert '2' haben.
- c. Das zweite Nibble MUSS hexadezimal die Anzahl Ziffern in *pin* codieren.
- d. Das $i + 2$ -te Nibble MUSS hexadezimal die i -te Ziffer von *pin* codieren.
- e. Alle anderen Nibble MÜSSEN den Wert 'F' besitzen.

8.1.8 Datum

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

(N008.120) K_Anwendungsspezifikation {K_Karte}

- a. 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.
- b. Das COS KANN weitere Codierungen für *date*

1. unterstützen oder
2. ablehnen.

8.2 Schlüsselmaterial (normativ)

8.2.1 Symmetrische Schlüssel

8.2.1.1 3DES-Schlüssel, Option_DES

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

(N008.200) *K_Anwendungsspezifikation* {*K_Karte*}, Option_DES

Der Wert von *3TDES_Key* MUSS ein Oktettstring mit 24 Oktetten sein.

(N008.300) *K_Anwendungsspezifikation* {*K_Karte*}, Option_DES

Der Wert von *3TDES_Key* MUSS beliebig wählbar sein, wobei für die Parität jedes Oktetts gilt: Falls die Parität der Oktette nicht korrekt ist, dann

- a. KANN das COS *3TDES_Key* akzeptieren.
- b. KANN das COS *3TDES_Key* ablehnen.

Hinweis (27): Aus Sicherheitsgründen sind die Teilschlüssel Ka, Kb und Kc 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 (28): Aus Sicherheitsgründen darf keiner der Teilschlüssel Ka, Kb oder Kc einen Wert haben, welcher einem schwachen oder halbschwachen DES-Schlüssel entspricht (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.

(N008.400) *K_Anwendungsspezifikation* {*K_Karte*}

Der Wert von *aes128Key* MUSS ein Oktettstring mit sechzehn Oktette sein.

(N008.500) *K_Anwendungsspezifikation* {*K_Karte*}

Der Wert von *aes128Key* MUSS beliebig wählbar sein.

8.2.1.3 AES-192-Schlüssel

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

(N008.520) *K_Anwendungsspezifikation* {*K_Karte*}

Der Wert von *aes192Key* MUSS ein Oktettstring mit 24 Oktette sein.

(N008.525) *K_Anwendungsspezifikation* {*K_Karte*}

Der Wert von *aes192Key* MUSS beliebig wählbar sein.

8.2.1.4 AES-256-Schlüssel

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

(N008.540) K_Anwendungsspezifikation {K_Karte}

Der Wert von *aes256Key* MUSS ein Oktettstring mit 32 Oktette sein.

(N008.545) K_Anwendungsspezifikation {K_Karte}

Der Wert von *aes256Key* MUSS beliebig wählbar sein.

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 7: Liste der Domainparameter einer elliptischen Kurve

Parameter	Bedeutung
p	Primzahl, welche die zugrunde liegende Gruppe F_p beschreibt
a	Erster Koeffizient der Weierstraßschen Gleichung
b	Zweiter Koeffizient der Weierstraßschen Gleichung
G	Ein Punkt auf der Kurve $E(F_p)$, Basispunkt
n	Ordnung des Basispunktes G in $E(F_p)$
h	Cofaktor von G in $E(F_p)$; wegen (N002.500) gilt $h = 1$ für alle Kurven, die ein COS unterstützen MUSS
L	siehe (N008.600)b
τ	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 Tabelle 7:

(N008.600) K_Anwendungsspezifikation {K_Karte}

domainParameter enthält

- alle Parameter aus Tabelle 7 und
- 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 hinzugefügt, da dieser Parameter in diesem Dokument vielfach verwendet wird. Allgemein gilt $L = \lceil \log_{256} p \rceil$. Wegen (N002.500) gilt:
 - $L = 32$ für brainpoolP256r1 und ansix9p256r1.
 - $L = 48$ für brainpoolP384r1 und ansix9p384r1.
 - $L = 64$ für brainpoolP512r1.
- eine Zahl τ , welche die Bitlänge von n angibt (siehe [BSI-TR-03111#Table 1.1]). Dieser Parameter wurde der „offiziellen“ Liste von Domainparametern hinzuge-

fügt, da dieser Parameter in diesem Dokument vielfach verwendet wird. Allgemein gilt $\tau = \text{ceiling}(\log_2 n)$. Wegen (N002.500) gilt:

1. $\tau = 256$ für brainpoolP256r1 und ansix9p256r1.
 2. $\tau = 384$ für brainpoolP384r1 und ansix9p384r1.
 3. $\tau = 512$ für brainpoolP512r1.
- d. 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 hinzugefügt, da in diesem Dokument Domainparameter vielfach per *OID* referenziert werden. *OID* MUSS so aus Tabelle 271 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 *privateEkcKey*.

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:

(N008.700) K_Anwendungsspezifikation {K_Karte}

Ein *privateRsaKey* MUSS ein Attribut Modulus n besitzen, dessen Länge gemäß (N002.100)a gewählt werden MUSS.

(N008.710) K_Anwendungsspezifikation {K_Karte}

Ein *privateRsaKey* MUSS ein Attribut *modulusLength* gemäß (N002.100)a besitzen.

(N008.800) K_Anwendungsspezifikation {K_Karte}

Ein *privateRsaKey* MUSS die im Folgenden genannten Attribute besitzen und es MUSS gelten:

- a. Falls das Attribut *keyAvailable* (siehe (N018.200)) den Wert
 1. False besitzt, dann MÜSSEN alle Attribute den Wert *AttributNotSet* besitzen.
 2. True besitzt, dann MÜSSEN alle Attribute ganze Zahlen mit den unten genannten Eigenschaften sein.
- b. Attribut Modulus n mit $n = p \cdot q$.
- c. Attribut öffentlicher Exponent e mit $\text{gcd}(e, (p-1)(q-1)) = 1$.
- d. Attribut privater Exponent d mit $e \cdot d \equiv 1 \pmod{\text{lcm}(p-1, q-1)}$.
- e. Attribut Primzahl p mit p ist prim und $p < q$.
- f. Attribut Primzahl q mit q ist prim.
- g. Attribut CRT Exponent zu d_p mit $d_p = d \pmod{p-1}$.
- h. Attribut CRT Exponent zu d_q mit $d_q = d \pmod{q-1}$.
- i. Attribut c mit $c = q^{-1} \pmod{p}$.

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:

(N008.900) K_Anwendungsspezifikation {K_Karte}

Ein *privateElcKey* MUSS ein Attribut *domainParameter* gemäß (N008.600) besitzen, die gemäß (N008.600)d per *OID* referenzierbar sind.

(N009.000) K_Anwendungsspezifikation {K_Karte}

Ein *privateElcKey* MUSS ein Attribut *d* besitzen, dessen Wert eine ganze Zahl im Intervall $[1, \text{domainParameter}.n - 1]$ sein MUSS.

8.2.4 Öffentlicher Schlüssel

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:

(N009.100) K_Anwendungsspezifikation {K_Karte}

Ein *publicRsaKey* MUSS ein Attribut *Modulus n* besitzen, dessen Länge gemäß (N002.100)a gewählt werden MUSS.

(N009.110) K_Anwendungsspezifikation {K_Karte}

Ein *publicRsaKey* MUSS ein Attribut *modulusLength* gemäß (N002.100)a besitzen.

(N009.200) K_Anwendungsspezifikation {K_Karte}

Ein *publicRsaKey* MUSS ein Attribut *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:

(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.

(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 Tabelle 144).

(N009.500) K_Anwendungsspezifikation {K_Karte}

Für *transportStatus* gilt folgender Wertebereich:

- a. regularPassword
- b. Leer-PIN
- c. Transport-PIN

(N009.600) K_COS

- a. Das COS MUSS alle Werte aus (N009.500) unterstützen.
- b. Ein COS KANN weitere Werte für *transportStatus*
 - 1. unterstützen oder
 - 2. ablehnen.

(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 Tabelle 8 verwendet werden.

(N009.710) K_COS

Wenn *oldSecret* oder *newSecret* in Tabelle 8 für eine Variante nicht erwähnt werden, so MUSS deren Inhalt im Rahmen von (N008.000) und (N008.100) beliebig wählbar sein.

Hinweis (29): Auch das Kommando RESET RETRY COUNTER ist in der Lage, den Transportschutz aufzuheben (siehe Use Cases in 14.6.5.1 und 14.6.5.3, sowie (N081.200)b).

Tabelle 8: Aufheben des Transportschutzes

Transportschutz	Parameter für Change Reference Data
regularPassword	P1='00'
Leer-PIN	P1='01', <i>oldSecret</i> = '', das heißt leerer String
Transport-PIN	P1='00'

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:

(N009.800) K_Anwendungsspezifikation {K_Karte}

Ein Ordner MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.

(N009.850) K_Anwendungsspezifikation {K_Karte}, Option_logische_Kanäle

Ein Ordner MUSS genau ein Attribut *shareable* vom Typ Boolean besitzen. Im Rahmen dieser Spezifikation MUSS der Wert des Attributes stets True sein, wodurch angezeigt wird, dass dieser Ordner in mehr als einem logischen Kanal als *currentFolder* verwendbar ist (siehe (N029.900)a).

(N009.900) K_Anwendungsspezifikation {K_Karte}

Ein Ordner MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.

(N010.000) K_Anwendungsspezifikation {K_Karte}

Ein Ordner MUSS eine (möglicherweise leere) Liste *children* mit Kindobjekten besitzen.

- a. Unter Berücksichtigung der maximalen Schachtelungstiefe (siehe (N020.400)) MÜSSEN Listenelemente der

1. Objekttypen

- | | |
|---|-----------------|
| 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) |

unterstützt werden.

2. Objekttypen Option_kontaktlose_Schnittstelle

- | | |
|--|---------------|
| i. Symmetrisches Kartenverbindungsobjekt | (siehe 8.6.2) |
|--|---------------|

unterstützt werden.

- b. Das COS KANN weitere Objekttypen als Listenelemente

1. unterstützen oder
2. ablehnen.

- c. 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)) denselben Wert besitzt.
3. deren Attribut *pwdIdentifier* (sofern vorhanden, siehe (N015.000)) denselben Wert besitzt.
4. deren Attribut *keyIdentifier* (sofern vorhanden, siehe (N016.400) und (N017.100)) denselben Wert besitzt.

(N010.005) K_COS

Ein Ordner MUSS die folgenden Kommandos unterstützen:

- a. ACTIVATE (siehe 14.2.1),
- b. DEACTIVATE (siehe 14.2.3),
- c. DELETE (siehe 14.2.4).
- d. FINGERPRINT (siehe 14.9.2),
- e. GET RANDOM (siehe 14.9.5, nur Option_logische_Kanäle),
- f. LOAD APPLICATION (siehe 14.2.5),
- g. SELECT (siehe 14.2.6),
- h. TERMINATE DF (siehe 14.2.8).

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

Hinweis (30): In (N010.000)a werden im Vergleich zur Vorgängerversion nun auch Applikationen (siehe 8.3.1.1) genannt, vergleiche auch (N020.200).

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, aber auch durch Benutzerverifikation oder Komponententhauthentisierung.

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:

(N010.100) K_Anwendungsspezifikation {K_Karte}

Eine Applikation ist eine Erweiterung von Ordner und MUSS deshalb den Anforderungen aus 8.3.1 genügen.

(N010.200) K_Anwendungsspezifikation {K_Karte}

Das Attribut *applicationIdentifier* (AID) MUSS ein Oktettstring sein.

- a. Die Länge von *applicationIdentifier* MUSS im Intervall [5, 16] liegen.
- b. Die Oktette in *applicationIdentifier* MÜSSEN beliebig wählbar sein.
- c. Ein COS KANN *applicationIdentifier*, die gegen diese Regeln verstoßen,
 - 1. akzeptieren oder
 - 2. ablehnen.

(N010.300) K_Anwendungsspezifikation {K_Karte}

Eine Applikation MUSS mindestens einen *applicationIdentifier* besitzen.

(N010.400) K_Anwendungsspezifikation {K_Karte}

- a. Eine Applikation DARF NICHT mehr als zwei *applicationIdentifier* besitzen.
- b. Ein COS KANN für Applikationen mehr als zwei *applicationIdentifier* erlauben.

8.3.1.2 Dedicated File

Ein Dedicated File (DF) ist ein Ordner, der nur mittels File Identifier 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:

(N010.500) K_Anwendungsspezifikation {K_Karte}

Ein DF ist eine Erweiterung von Ordner und MUSS deshalb den Anforderungen aus 8.3.1 genügen.

(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 (siehe 8.3.1.1) und DF (siehe 8.3.1.2) gesehen. Dementsprechend gelten für ein ADF folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

(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:

(N010.800) K_Anwendungsspezifikation {K_Karte}

Eine Datei MUSS genau ein Attribut vom Typ *fileIdentifier* (siehe 8.1.1) besitzen.

(N010.900) K_Anwendungsspezifikation {K_Karte}

Eine Datei MUSS eine Liste mit Elementen vom Typ *shortFileIdentifier* (siehe 8.1.2) unterstützen. Diese Liste ist entweder

- leer (die Datei hat keinen *shortFileIdentifier*) oder
- enthält genau ein Element (die Datei hat genau einen *shortFileIdentifier*).

(N011.000) K_Anwendungsspezifikation {K_Karte}

Eine Datei MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.

(N011.050) K_Anwendungsspezifikation {K_Karte}, Option_logische_Kanäle

Eine Datei MUSS genau ein Attribut *shareable* vom Typ Boolean besitzen. Im Rahmen dieser Spezifikation MUSS der Wert des Attributes stets True sein, wodurch angezeigt wird, dass diese Datei in mehr als einem logischen Kanal als *currentEF* verwendbar ist (siehe (N029.900)m)].

(N011.100) K_Anwendungsspezifikation {K_Karte}

Eine Datei MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.

(N011.200) K_Anwendungsspezifikation {K_Karte}

- a. Eine Datei MUSS ein Attribut *flagTransactionMode* vom Typ Boolean besitzen.
- b. Ein COS KANN pro Datei ein solches Attribut individuell verwalten.
- c. Ein COS KANN dieses Attribut für alle Dateien implizit auf True setzen.

(N011.300) K_Anwendungsspezifikation {K_Karte}

- a. Eine Datei MUSS ein Attribut *flagChecksum* vom Typ Boolean besitzen.
- b. Ein COS KANN pro Datei ein solches Attribut individuell verwalten.
- c. Ein COS KANN dieses Attribut für alle Dateien implizit auf True setzen.

*Hinweis (31): Das Setzen des Flags *flagTransactionMode* verringert die Performanz bei Schreibzugriffen.*

*Hinweis (32): 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:

(N011.400) K_Anwendungsspezifikation {K_Karte}

Ein transparentes EF ist eine Erweiterung von Datei und MUSS deshalb den Anforderungen aus 8.3.2 genügen.

(N011.500) K_Anwendungsspezifikation {K_Karte}

- a. Ein transparentes EF MUSS genau ein Attribut *numberOfOctet* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 32.768] sein MUSS.
- b. Ein COS KANN *numberOfOctet*, falls es außerhalb des Intervalls liegt,
 - 1. ablehnen oder
 - 2. akzeptieren.

(N011.510) K_Anwendungsspezifikation {K_Karte}

Ein transparentes EF MUSS genau ein Attribut *positionLogicalEndOfFile* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [0, *numberOfOctet*] sein MUSS.

(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*.

(N011.700) K_COS

Die Größe einer „data unit“ in *body* MUSS 1 Oktett betragen.

(N011.800) K_COS

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.

(N011.900) K_COS

Ein transparentes EF MUSS die folgenden Kommandos unterstützen:

- a. Bevor eines der folgenden Kommandos in der Lage ist mit dem Oktettstring zu arbeiten, ist das transparente EF zu selektieren. Dies geschieht entweder mittels SELECT-Kommando (siehe 14.2.6.13 und 14.2.6.14), oder mittels Short File Identifier, der als Parameter dem zugreifenden Kommando mitgegeben wird.
 1. ERASE BINARY (siehe 14.3.1),
 2. READ BINARY (siehe 14.3.2),
 3. SET LOGICAL EOF (siehe 14.3.4),
 4. UPDATE BINARY (siehe 14.3.5).
 5. WRITE BINARY (siehe 14.3.6).
- b. Bevor eines der folgenden Kommandos in der Lage ist mit dem Oktettstring zu arbeiten, ist das transparente EF zu selektieren. Dies geschieht mittels SELECT-Kommando (siehe 14.2.6.13 und 14.2.6.14).
 1. ACTIVATE (siehe 14.2.1),
 2. DEACTIVATE (siehe 14.2.3),
 3. DELETE (siehe 14.2.4).
 4. TERMINATE (siehe 14.2.9).

(N012.000) K_COS

Ein transparentes EF KANN weitere Kommandos

- a. unterstützen oder
- b. ablehnen.

Hinweis (33): 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 ERASE BINARY, 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:

(N012.100) K_Anwendungsspezifikation {K_Karte}

Ein strukturiertes EF ist eine Erweiterung von Datei und MUSS deshalb den Anforderungen aus 8.3.2 genügen.

(N012.200) K_Anwendungsspezifikation {K_Karte}

- a. Ein strukturiertes EF MUSS ein Attribut *recordList* vom Typ Liste mit Elementen vom Typ Rekord besitzen. Die Anzahl der Listenelemente MUSS im Intervall [0, 254] liegen.
- b. Ein COS KANN eine größere Anzahl von Listenelementen
 - 1. zulassen oder
 - 2. ablehnen.

(N012.300) K_COS

Das *i*-te Element in *recordList* MUSS die Rekordnummer *i* besitzen.

(N012.400) K_Anwendungsspezifikation {K_Karte}

- a. Ein strukturiertes EF MUSS genau ein Attribut *maximumNumberOfRecords* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 254] sein MUSS.
- b. Ein COS KANN *maximumNumberOfRecords*, falls dessen Wert außerhalb dieses Intervalls liegt,
 - 1. akzeptieren, oder
 - 2. ablehnen.

(N012.500) K_Anwendungsspezifikation {K_Karte}

- a. Ein strukturiertes EF MUSS genau ein Attribut *maximumRecordLength* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 255] sein MUSS.
- b. Ein COS KANN *maximumRecordLength*, falls dessen Wert außerhalb dieses Intervalls liegt,
 - 1. akzeptieren oder
 - 2. ablehnen.

(N012.600) K_Anwendungsspezifikation {K_Karte}

Ein strukturiertes EF MUSS genau ein Attribut *flagRecordLifeCycleStatus* vom Typ Boolean besitzen. Falls dieses Attribut

- a. True ist, dann MÜSSEN alle Elemente in *recordList* einen *lifeCycleStatus* besitzen.
- b. False ist, dann DARF KEIN Element in *recordList* einen *lifeCycleStatus* besitzen.

(N012.700) K_COS

Ein strukturiertes EF MUSS die folgenden Kommandos unterstützen:

- a. Bevor eines der folgenden Kommandos in der Lage ist spezifikationsgemäß zu arbeiten, ist das strukturierte EF zu selektieren. Dies geschieht entweder mittels SELECT-Kommando (siehe 14.2.6.13 und 14.2.6.14), oder mittels Short File Identifier, der als Parameter dem zugreifenden Kommando mitgegeben wird.
 - 1. ACTIVATE RECORD (siehe 14.4.1),
 - 2. APPEND RECORD (siehe 14.4.2),
 - 3. DEACTIVATE RECORD (siehe 14.4.3),
 - 4. DELETE RECORD (siehe 14.4.4),
 - 5. ERASE RECORD (siehe 14.4.5),
 - 6. READ RECORD (siehe 14.4.6),
 - 7. SEARCH RECORD (siehe 14.4.7),

- 8. UPDATE RECORD (siehe 14.4.8).
 - b. Bevor eines der folgenden Kommandos in der Lage ist spezifikationsgemäß zu arbeiten, ist das strukturierte EF zu selektieren. Dies geschieht mittels SELECT-Kommando (siehe 14.2.6.13 und 14.2.6.14).
 - 1. ACTIVATE (siehe 14.2.1),
 - 2. DEACTIVATE (siehe 14.2.3),
 - 3. DELETE (siehe 14.2.4).
 - 4. TERMINATE (siehe 14.2.9).
- (N012.800) K_COS
Ein strukturiertes EF KANN weitere Kommandos
- a. unterstützen oder
 - b. ablehnen.

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:

(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.

(N013.000) K_Anwendungsspezifikation {K_Karte}

- a. Ein linear variables EF MUSS genau ein Attribut *numberOfOctet* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [1, 64.770] sein MUSS.
- b. Ein COS KANN *numberOfOctet*, falls es außerhalb des Intervalls liegt,
 - 1. akzeptieren oder
 - 2. ablehnen.

(N013.100) K_COS

Wird mittels APPEND RECORD der Liste ein neuer *record* hinzugefügt, so MUSS der neue *record* am Ende der Liste eingefügt werden.

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:

(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.

(N013.300) K_Anwendungsspezifikation {K_Karte}

Jeder *record* in *recordList* MUSS *maximumRecordLength* Oktette besitzen.

(N013.400) K_COS

Wird mittels Append Record der Liste ein neuer *record* hinzugefügt, so MUSS der neue *record* am Ende der Liste eingefügt werden.

(N013.410) K_Anwendungsspezifikation {K_Karte}

Das Produkt aus *maximumNumberOfRecords* und *maximumRecordLength* MUSS kleiner gleich 64.770 sein (vergleiche auch (N013.000)).

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:

(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.

(N013.600) K_Anwendungsspezifikation {K_Karte}

Jeder *record* in *recordList* MUSS *maximumRecordLength* Oktette besitzen.

(N013.700) K_COS

Wird mittels Append Record der Liste ein neuer *record* hinzugefügt, so MUSS der neue *record* am Anfang der Liste eingefügt werden.

(N013.800) K_Anwendungsspezifikation {K_Karte}

Zyklische Elementary Files MÜSSEN für das Attribut *flagRecordLifeCycleStatus* den Wert False verwenden.

(N013.810) K_Anwendungsspezifikation {K_Karte}

Das Produkt aus *maximumNumberOfRecords* und *maximumRecordLength* MUSS kleiner gleich 64.770 sein (vergleiche auch (N013.000)).

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 Abbildung 1) in den Antwortdaten eines Select-Kommandos (siehe (N048.300)a) sichtbar. Für die Codierung der File Control Parameter gilt:

(N013.900) K_COS

Die File Control Parameter (FCP) MÜSSEN gemäß DER-TLV in einem DO_FCP codiert werden, welches ein Tag = '62' besitzen MUSS.

(N014.000) K_COS

'80': Falls das File vom Typ transparentes EF (siehe 8.3.2.1) oder linear variables EF (siehe 8.3.2.2.1) ist, genau dann MUSS DO_FCP ein DO_Size enthalten. Es gilt:

a. DO_Size MUSS ein Tag = '80' besitzen.

b. Das Wertfeld KANN eine beliebige Länge besitzen.

- c. Wird der Oktettstring des Wertfeldes mittels OS2I(...) konvertiert, dann MUSS diese Zahl gleich *numberOfOctet* (siehe (N011.500), bzw. (N013.000)) sein.

(N014.100) K_COS

'82': DO_FCP MUSS ein DO_FileDescriptor enthalten. Falls File vom Typ

- a. Ordner ist, dann gilt:

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.

- b. Transparentes EF ist, dann gilt:

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.

- c. Strukturiertes EF ist, dann gilt:

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, falls File vom Typ linear fixes EF ist.
 - ii. {'04', '44'} besitzen, falls File vom Typ linear variables EF ist.
 - iii. {'06', '46'} besitzen, falls 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.

(N014.200) K_COS

'83': Falls das File ein Attribut vom Typ *fileIdentifier* gemäß 8.1.1 besitzt, genau dann MUSS DO_FCP ein DO_FID enthalten. Es gilt:

DO_FID = '83 02 I2OS(*fileIdentifier*, 2)'.

(N014.300) K_COS

'84': Falls das File Attribute vom Typ *applicationIdentifier* gemäß (N010.200) besitzt, dann MUSS DO_FCP jedes dieser Attribute enthalten. Jedes dieser Attribute MUSS als Wertfeld in einem DO_AID codiert werden. Es gilt:

DO_AID = '84 || I2OS(OctetLength(*applicationIdentifier*, 1) || *applicationIdentifier*'.

(N014.400) K_COS

'88': Falls das File vom Typ Datei ist, genau dann MUSS DO_FCP ein DO_SFI enthalten. Falls die Datei

- a. ein Attribut *shortFileIdentifier* gemäß 8.1.2 besitzt, gilt:

DO_SFI = '88 01 I2OS(8 *shortFileIdentifier*, 1)'.

- b. kein Attribut *shortFileIdentifier* besitzt, gilt:

DO_SFI = '88 00'.

(N014.500) K_COS

'8A': Der physikalische Wert des Attributes *lifeCycleStatus* (siehe 8.1.3) MUSS in DO_FCP enthalten sein. Es MUSS als Wertfeld in einem DO_LCS mit Tag = '8A' codiert werden. Der Wert von *lifeCycleStatus* MUSS in einem Oktett gemäß [ISO/IEC 7816-4#Tabelle 14] codiert werden. Das bedeutet

- a. „Operational state (active)“ MUSS aus der Menge {'05', '07'} stammen.
- b. „Operational state (deactivated)“ MUSS aus der Menge {'04', '06'} stammen.
- c. „Termination state“ MUSS aus der Menge {'0C', '0D', '0E', '0F'} stammen.

(N014.600) K_COS

'8F': Falls das File vom Typ strukturiertes EF ist und das Attribut *flagRecordLifeCycleStatus* hat den Wert

- a. False, dann KANN DO_FCP
 - 1. ein DO_ProfileIdentifier = '8F 01 00' enthalten.
 - 2. kein DO mit Tag = '8F' enthalten.
- b. True, genau dann MUSS DO_FCP ein DO_ProfileIdentifier = '8F 01 XX' enthalten. OS2I('XX') MUSS eine ungerade Zahl kleiner als 128 sein.

(N014.700) K_COS

'C5': Falls das File vom Typ transparentes EF (siehe 8.3.2.1) ist, genau dann MUSS DO_FCP ein DO_ReadSize enthalten. Es gilt:

- a. DO_ReadSize MUSS ein Tag = 'C5' besitzen.
- b. Das Wertfeld KANN eine beliebige Länge besitzen.
- c. Wird der Oktettstring des Wertfeldes mittels OS2I(...) konvertiert, dann MUSS diese Zahl gleich *positionLogicalEndOfFile* (siehe (N011.510)) sein.

(N014.800) K_COS

In DO_FCP KÖNNEN weitere Datenobjekte enthalten sein. Deren Codierung und Bedeutung KANN herstellerspezifisch sein.

(N014.900) K_COS

Für die Reihenfolge der Datenobjekte in DO_FCP gilt:

- a. Die Reihenfolge aller Datenobjekte ist herstellerspezifisch.
- b. Falls in DO_FCP vorhanden, dann MÜSSEN 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 sein.

8.4 Reguläres-Passwort (normativ)

Hinweis (34): Mit dieser Version des Dokumentes wird der Objekttyp "Multireferenz-Passwort" eingeführt (siehe 8.5). 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 (35): 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:

(N015.000) K_Anwendungsspezifikation {K_Karte}

- a. Ein Reguläres-Passwort MUSS genau ein Attribut *pwdIdentifier* besitzen, dessen Wert eine ganz Zahl aus dem Intervall [0, 31] sein MUSS.
- b. Ein COS KANN weitere Werte für *pwdIdentifier*
 1. unterstützen oder
 2. ablehnen.

(N015.050) K_Anwendungsspezifikation {K_Karte}

Eine Reguläres-Passwort MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.

(N015.100) K_Anwendungsspezifikation {K_Karte}

Ein Reguläres-Passwort MUSS genau ein Attribut *accessRules* vom Typ *interface-DependentAccessRules* (siehe 8.1.4) besitzen.

(N015.200) K_Anwendungsspezifikation {K_Karte}

Ein Reguläres-Passwort MUSS genau ein Attribut *secret* vom Typ *pin* (siehe 8.1.7) besitzen.

(N015.300) K_Anwendungsspezifikation {K_Karte}

- a. Das Reguläres-Passwort MUSS genau ein Attribut *minimumLength* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [4, 12] sein MUSS.
- b. Ein COS KANN weitere Werte für die *minimumLength*
 1. unterstützen oder
 2. ablehnen.

(N015.310) K_Anwendungsspezifikation {K_Karte}

- a. Das Reguläres-Passwort MUSS genau ein Attribut *maximumLength* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [*minimumLength*, 12] sein MUSS.
- b. Ein COS KANN weitere Werte für die *maximumLength*

1. unterstützen oder
2. ablehnen.

(N015.400) K_Anwendungsspezifikation {K_Karte}

- a. Ein Reguläres-Passwort MUSS genau ein Attribut *startRetryCounter* besitzen, dessen Wert eine ganze Zahl im Intervall [1, 15] sein MUSS.
- b. Ein COS KANN weitere Werte für *startRetryCounter*
 1. unterstützen oder
 2. ablehnen.

(N015.500) K_Anwendungsspezifikation {K_Karte}

- a. Ein Reguläres-Passwort MUSS genau ein Attribut *retryCounter* besitzen, dessen Wert eine ganze Zahl im Intervall [0, 15] sein MUSS.
- b. Ein COS KANN weitere Werte für *retryCounter*
 1. unterstützen oder
 2. ablehnen.

(N015.600) K_Anwendungsspezifikation {K_Karte}

Ein Reguläres-Passwort MUSS genau ein Attribut *transportStatus* gemäß 8.2.5 besitzen.

(N015.610) K_COS {K_Karte}

Falls das Attribut *transportStatus* eines regulären Passwortobjektes vom Zustand "Leer-PIN" in den Zustand "regularPassword" wechselt, dann MUSS auf eine herstellerspezifische Art und Weise sichergestellt sein, dass der Use Case "Setzen eines Benutzergeheimnisses" gemäß 14.6.1.2 für dieses reguläre Passwortobjekt und alle auf dieses reguläre Passwortobjekt verweisenden Multireferenz-Passwortobjekte nicht mehr ausführbar ist.

(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).

(N015.800) K_Anwendungsspezifikation {K_Karte}

Ein Reguläres-Passwort MUSS genau ein Attribut *startSsecList* besitzen

- a. Die Liste
 1. MUSS mindestens ein Element und
 2. DARF NICHT mehr als vier Elemente enthalten.
- b. Ein COS KANN Listen mit mehr als vier Elementen
 1. unterstützen oder
 2. ablehnen.
- c. Jedes Listenelement MUSS
 1. genau einen *seldentifier* gemäß (N007.900) und
 2. genau eine ganze Zahl *startSsec* enthalten. *startSsec* MUSS aus dem Intervall [1, 250] sein oder den Wert "unendlich" repräsentieren.
- d. Ein COS KANN weitere Werte für *startSsec*

1. unterstützen oder
2. ablehnen.

(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 Zurücksetzen eines abgelaufenen *retryCounter* ermöglicht.

(N016.000) K_Anwendungsspezifikation {K_Karte}

- a. Ein Reguläres-Passwort MUSS ein Attribut *pukUsage* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [0, 15] sein MUSS.
- b. Ein COS KANN weitere Werte für *pukUsage*
 1. unterstützen oder
 2. ablehnen.

(N016.100) K_externeWelt {K_Karte}

- a. Die Ziffernfolgen *secret* und *PUK* MÜSSEN an der Schnittstelle Interpreter (siehe Abbildung 1) stets als Format-2-PIN-Block (siehe (N008.100)) übertragen werden
- b. Ein COS KANN weitere Übertragungsformate
 1. unterstützen oder
 2. ablehnen.

(N016.200) K_COS

Ein Passwort MUSS die folgenden Kommandos unterstützen:

- a. 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.
 1. CHANGE REFERENCE DATA (siehe 14.6.1),
 2. DISABLE VERIFICATION REQUIREMENT (siehe 14.6.2),
 3. ENABLE VERIFICATION REQUIREMENT (siehe 14.6.3),
 4. GET PIN STATUS (siehe 14.6.4),
 5. RESET RETRY COUNTER (siehe 14.6.5),
 6. VERIFY (siehe 14.6.6).
- b. Bevor eins 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.
 1. ACTIVATE (siehe 14.2.1),
 2. DEACTIVATE (siehe 14.2.3),
 3. DELETE (siehe 14.2.4).
 4. TERMINATE (siehe 14.2.9).

(N016.210) K_externeWelt {K_Karte}

Die externe Welt DARF ein reguläres Passwort NICHT löschen (Use Case entsprechend 14.2.4.4), solange es Multireferenz-Passwortobjekte gibt, die mittels ihres Attributes *passwordReference* (siehe (N016.320)f) auf dieses reguläre Passwortobjekt verweisen.)

(N016.300) K_COS

Ein Reguläres-Passwort KANN weitere Kommandos

- a. unterstützen oder
- b. ablehnen.

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 14.6.6) geändert.

Hinweis (36): 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:

(N016.320) K_Anwendungsspezifikation {K_Karte}

Folgende Attribute lassen sich für ein Multireferenz-Passwort individuell festlegen, das heißt diese Attribute werden nicht gemeinsam mit anderen Regulären-Passwörtern oder anderen Multireferenz-Passwörtern genutzt:

- a. *pwdIdentifier*: Ein Multireferenz-Passwort MUSS genau ein Attribut *pwdIdentifier* besitzen. Für dieses Attribut gelten die in (N015.000) genannten Anforderungen.
- b. *lifeCycleStatus*: Eine Multireferenz-Passwort MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
- c. *accessRules*: Ein Multireferenz-Passwort MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
- d. *startSsecList*: Ein Multireferenz-Passwort MUSS genau ein Attribut *startSsecList* besitzen. Für diese Liste gelten die in (N015.800) genannten Anforderungen.
- e. *flagEnabled*: Ein Multireferenz-Passwort MUSS genau ein Attribut *flagEnabled* besitzen. Für diese Liste gelten die in (N015.700) genannten Anforderungen.
- f. *passwordReference*:
 - 1. Dieses Attribut MUSS eine Referenz auf ein Passwort enthalten.
 - 2. Sei *objectFolder* der Ordner, welcher dieses Multireferenz-Passwort enthält, dann MUSS gelten:
 - i. Die Funktion *pwd = SearchPwd(objectFolder, passwordReference)* DARF NICHT den Fehler *pwdNotFound* zurückmelden.

- ii. *pwd* MUSS ein Reguläres-Passwort sein (keine Kettenbildung).
- iii. Die Attribute, welche dieses Multireferenz-Passwort gemeinsam mit einem Regulären-Passwort nutzt, MÜSSEN dem Passwort *pwd* entnommen werden. Mit anderen Worten: Die Attribute, welches das Multireferenz-Passwort nicht selber speichert, steuert *pwd* bei.

(N016.325) K_Anwendungsspezifikation {K_Karte}

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

- a. *secret* MUSS mit *pwd* gemeinsam genutzt werden.
- b. *minimumLength* MUSS mit *pwd* gemeinsam genutzt werden.
- c. *maximumLength* MUSS mit *pwd* gemeinsam genutzt werden.
- d. *startRetryCounter* MUSS mit *pwd* gemeinsam genutzt werden.
- e. *retryCounter* MUSS mit *pwd* gemeinsam genutzt werden.
- f. *transportStatus* MUSS mit *pwd* gemeinsam genutzt werden.
- g. *PUK* MUSS mit *pwd* gemeinsam genutzt werden.
- h. *pukUsage* MUSS mit *pwd* gemeinsam genutzt werden.

(N016.330) K_COS

Ein Multireferenz-Passwort MUSS dieselben Kommandos unterstützen wie ein Reguläres-Passwort (siehe (N016.200)).

(N016.335) K_COS

Ein Multireferenz-Passwort KANN weitere Kommandos

- a. unterstützen oder
- b. ablehnen.

Einem Multireferenz-Passwort ist ein weiteres, kanalspezifisches Attribut *securityStatus-EvaluationCounter* (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 14.6.6) 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).
2. Mit persistent gespeichertem Geheimnis (Schlüssel) zur gegenseitigen Authentisierung bei gleichzeitiger Aushandlung von Sessionkeys (siehe 15.4.1).
3. Als Sessionkeys zur Sicherstellung einer vertraulichen Kommunikation.

4. Als Sessionkeys 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 (N084.400)c)

Öffentliche Schlüssel werden in diesem Dokument zu folgenden Zwecken eingesetzt:

9. Prüfen elektronischer Signaturen beim Import von Zertifikaten (s. (N095.900))
10. Prüfen von Signaturen im Rahmen von Rollenauthentisierungen (s. (N084.400))
11. Transportsicherung von Sessionkey Material (s. (N086.900)d)
12. Verschlüsseln von Daten (s. 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:

(N016.400) K_Anwendungsspezifikation {K_Karte}

- a. Ein symmetrisches Authentisierungsobjekt MUSS ein Attribut *keyIdentifier* besitzen, dessen Wert eine ganze Zahl aus dem Intervall [2, 28] sein MUSS.
- b. Ein COS KANN weitere Werte für *keyIdentifier*
 1. unterstützen oder
 2. ablehnen.

(N016.450) K_Anwendungsspezifikation {K_Karte}

Eine symmetrisches Authentisierungsobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.

(N016.500) K_Anwendungsspezifikation {K_Karte}

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

(N016.590) K_Anwendungsspezifikation {K_Karte}

- a. Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *keyType* aus der Menge {3TDES (Option_DES), AES-128, AES-192, AES-256} besitzen.
- b. Ein COS KANN weitere Werte für *keyType*
 1. unterstützen oder
 2. ablehnen.

(N016.600) K_Anwendungsspezifikation {K_Karte}

Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *encKey* gemäß 8.2.1 besitzen.

- (N016.700) K_Anwendungsspezifikation {K_Karte}
Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut *macKey* gemäß 8.2.1 besitzen.
- (N016.705) K_Anwendungsspezifikation {K_Karte}
Die Attribute *encKey* und *macKey* MÜSSEN passend zum Attribut *keyType* gewählt werden.
- (N016.710) K_Anwendungsspezifikation {K_Karte}
Falls *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.
- (N016.800) K_Anwendungsspezifikation
Ein symmetrisches Authentisierungsobjekt MUSS genau ein Attribut vom Typ *algorithmIdentifier* besitzen, welches angibt, für welchen Zweck es verwendbar ist.
- {K_Karte}
Für symmetrische Authentisierungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
 - aesSessionkey4SM*, (siehe, (N084.410)a)
 - desSessionkey4SM*, (siehe, (N084.410)b, Option_DES)} sein (siehe Tabelle 267).
 - {K_Karte}, Option_Kryptobox
Für symmetrische Authentisierungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {
 - aesSessionkey4TC*, (siehe (N084.402)a, (N086.902)a)
 - desSessionkey4TC* (siehe (N084.402)b, (N086.902)b, Option_DES)} sein (siehe Tabelle 267).
 - Ein COS KANN weitere Werte für *algorithmIdentifier*
 - unterstützen oder
 - ablehnen.
- (N016.820) K_Anwendungsspezifikation {K_Karte}, Option_Kryptobox
Falls *algorithmIdentifier* Element der Menge {*aesSessionkey4TC*, *desSessionkey4TC*} ist, genau dann MUSS ein symmetrisches Authentisierungsobjekt ein Attribut *accessRulesSessionkeys* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.
- (N016.900) K_COS
Ein symmetrisches Authentisierungsobjekt MUSS die folgenden Kommandos unterstützen:
- 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.
 - EXTERNAL AUTHENTICATE (siehe 14.7.1.1 und 14.9.9.4),
 - GENERAL AUTHENTICATE (siehe 14.7.2.3.1 und 14.9.9.4),
 - INTERNAL AUTHENTICATE (siehe 14.7.4.1 und 14.9.9.2),
 - MUTUAL AUTHENTICATE (siehe 14.7.1.2 und 14.9.9.6),

- b. 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.

1. ACTIVATE (siehe 14.2.1)
2. DEACTIVATE (siehe 14.2.3),
3. DELETE (siehe 14.2.4).
4. GET SECURITY STATUS KEY (siehe 14.7.3.1),
5. TERMINATE (siehe 14.2.9).

(N017.000) K_COS

Ein symmetrisches Authentisierungsobjekt KANN weitere Kommandos

- a. unterstützen oder
- b. ablehnen.

Hinweis (37): 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:

(N017.020) K_Anwendungsspezifikation {K_Karte},
Option_kontaktlose_Schnittstelle, Option_PACE_PCD

- a. Ein symmetrisches Kartenverbindungsobjekt MUSS ein Attribut *keyIdentifier* besitzen, dessen Wert eine ganz Zahl aus dem Intervall [2, 28] sein MUSS.
- b. Ein COS KANN weitere Werte für *keyIdentifier*
 1. unterstützen oder
 2. ablehnen.

(N017.024) K_Anwendungsspezifikation {K_Karte},
Option_kontaktlose_Schnittstelle, Option_PACE_PCD

Eine symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.

(N017.026) K_Anwendungsspezifikation {K_Karte},
Option_kontaktlose_Schnittstelle, Option_PACE_PCD

Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.

(N017.028) K_Anwendungsspezifikation {K_Karte},
Option_kontaktlose_Schnittstelle, Option_PACE_PCD

Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut *can* vom Typ *pin* gemäß (N008.000) besitzen.

(N017.030) K_Anwendungsspezifikation {K_Karte},

Option_kontaktlose_Schnittstelle, Option_PACE_PCD

Ein symmetrisches Kartenverbindungsobjekt MUSS genau ein Attribut vom Typ *algorithmIdentifier* besitzen, welches angibt, für welchen Zweck es verwendbar ist.

a. Option_kontaktlose_Schnittstelle: Für symmetrische Kartenverbindungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {

1. id-PACE-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
2. id-PACE-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
3. id-PACE-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)

} sein.

b. Option_PACE_PCD: Für symmetrische Kartenverbindungsobjekte MUSS der Wert von *algorithmIdentifier* Element der Menge {

1. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
2. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
3. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)

} sein.

c. Ein COS KANN weitere Werte für *algorithmIdentifier*

1. unterstützen oder
2. ablehnen.

(N017.031) K_Anwendungsspezifikation {K_Karte},

Option_PACE_PCD

Falls *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.

(N017.032) K_COS,

Option_kontaktlose_Schnittstelle, Option_PACE_PCD

Ein symmetrisches Kartenverbindungsobjekt

a. MUSS die folgenden Kommandos unterstützen:

1. Bevor eines der folgenden Kommandos in der Lage ist mit dem symmetrischen Kartenverbindungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.7 beschriebenen Use Cases.

i. GENERAL AUTHENTICATE (siehe 14.7.2 und 14.9.9.7).

2. 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.

i. ACTIVATE (siehe 14.2.1),

ii. DEACTIVATE (siehe 14.2.3),

- iii. DELETE (siehe 14.2.4).
- iv. TERMINATE (siehe 14.2.9).

b. KANN weitere Kommandos

- 1. unterstützen oder
- 2. ablehnen.

Hinweis (38): 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 (39): 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. Andererseits wird zum Aufbau einer IPSec-Verbindung gegebenenfalls ein privates Schlüsselobjekt verwendet, welches 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:

(N017.100) K_Anwendungsspezifikation {K_Karte}

- a. Ein privates Schlüsselobjekt MUSS ein Attribut *keyIdentifier* besitzen, dessen Wert eine ganz Zahl aus dem Intervall [2, 28] sein MUSS.
- b. Ein COS KANN weitere Werte für *keyIdentifier*
 - 1. unterstützen oder
 - 2. ablehnen.

(N017.150) K_Anwendungsspezifikation {K_Karte}

Ein privates Schlüsselobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.

(N017.200) K_Anwendungsspezifikation {K_Karte}

Ein privates Schlüsselobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.

(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. Der ursprüngliche Text forderte eine Personalisierung von privaten Schlüsseln, die komplett durch das Kommando GENERATE ASYMMETRIC KEY PAIR abgedeckt ist.

(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.

- a. Für die Länge der Liste gilt: Die Liste MUSS mindestens ein Element und DARF NICHT mehr als vier Elemente enthalten.
- b. Jedes Listenelement MUSS
 1. genau einen *seldentifier* 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.
- c. Die Menge *setAlgorithmIdentifier* DARF NICHT mehr als sechs Elemente enthalten.
- d. Das COS KANN
 1. Listen des Typs *listAlgorithmIdentifier*, welche mehr Elemente enthalten als nach (N017.400)a gefordert,
 - i. unterstützen, oder
 - ii. ablehnen.
 2. Listen des Typs *setAlgorithmIdentifier*, welche andere Elemente enthalten als nach (N017.400)b.3 gefordert,
 - i. unterstützen, oder
 - ii. ablehnen.
 3. Listen des Typs *setAlgorithmIdentifier*, welche mehr Elemente enthalten als nach (N017.400)c gefordert,
 - i. unterstützen, oder
 - ii. ablehnen.

(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.

(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.

(N017.600) K_Anwendungsspezifikation {K_Karte}

Wertebereich von *algorithmIdentifier*

- a. Ist das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ
1. *privateRsaKey* MUSS zum Zwecke der Authentisierung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 268): {
 - i. *rsaClientAuthentication*, (siehe 14.7.4.1 und (N086.900)b)
 - ii. *rsaRoleAuthentication*, (siehe 14.7.4.1 und (N086.900)c, Option_RSA_CVC)
 - iii. *rsaSessionkey4SM*, (siehe 14.7.4.1 und (N086.900)d, Option_DES)
 - iv. *rsaSessionkey4TC*, ((N084.402)c,(N086.902)c, Option_Kryptobox, Option_DES)}
 2. *privateElcKey* MUSS zum Zwecke der Authentisierung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 268): {
 - i. *elcAsynchronAdmin*, (siehe 14.7.2.5 und (N085.068))
 - ii. *elcRoleAuthentication*, (siehe 14.7.4.1 und (N086.900)a)
 - iii. *elcSessionkey4SM*, (siehe (N085.054))
 - iv. *elcSessionkey4TC*, (siehe (N085.056), Option_Kryptobox)}
- b. Ein COS KANN weitere Werte für *algorithmIdentifier*
1. unterstützen oder
 2. ablehnen.

(N017.610) K_Anwendungsspezifikation {K_Karte}, Option_DES, Option_RSA_CVC
Zusammenhang zwischen Modulslänge und Card-to-Card-Algorithmen:

- a. Die *algorithmIdentifier* *rsaRoleAuthentication*, *rsaSessionkey4SM* und *rsaSessionkey4TC* DÜRFEN einem privaten Schlüssel NICHT zugewiesen werden, wenn dessen Modulslänge ungleich 2048 bit ist.
- b. Das COS KANN die Algorithmen *rsaRoleAuthentication*, *rsaSessionkey4SM* und *rsaSessionkey4TC* für andere Modulslängen
 1. unterstützen oder
 2. ablehnen.

(N017.620) K_Anwendungsspezifikation {K_Karte}
Zusammenhang zwischen Kurvenparametern und Card-to-Card-Algorithmen:

- a. Die *algorithmIdentifier* *elcRoleAuthentication*, *elcSessionkey4SM* und *elcSessionkey4TC* DÜRFEN einem privaten Schlüssel NICHT zugewiesen werden, wenn dessen *domainParameter* nicht in der Menge {brainpoolP256r1, brainpoolP384r1, brainpoolP512r1} enthalten ist.
- b. Das COS KANN die Algorithmen *elcRoleAuthentication*, *elcSessionkey4SM* und *elcSessionkey4TC* für andere *domainParameter*
 1. unterstützen oder
 2. ablehnen.

(N017.700) K_COS

Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels der in 14.9.9 beschriebenen Use Cases. Ist einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus der in (N017.600) genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:

- a. EXTERNAL AUTHENTICATE (siehe (N084.400)c und 14.9.9.3)
- b. GENERAL AUTHENTICATE (siehe 15.4.4),
- c. INTERNAL AUTHENTICATE (siehe 14.7.4.1 und 14.9.9.3),
- d. PSO Compute Digital Signature (siehe (N088.600)c und 14.9.9.9)

(N017.800) Dieser Punkt ist absichtlich leer.

(N017.900) K_Anwendungsspezifikation {K_Karte}

Wertebereich von *algorithmIdentifier*

- a. Ist das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ
 - 1. *privateRsaKey* MUSS zum Zwecke der Entschlüsselung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 269): {
 - i. *rsaDecipherPKCS1_V1_5*, (siehe 14.8.3.1, 14.8.6.1 und 14.8.6.2)
 - ii. *rsaDecipherOaep*, (siehe 14.8.3.1, 14.8.6.1 und 14.8.6.2)
 - }.
 - 2. *privateElcKey* MUSS zum Zwecke der Entschlüsselung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 269): {
 - i. *elcSharedSecretCalculation*, (s. 14.8.3.2, 14.8.6.2, 14.8.6.3, 14.8.6.4)
 - }.
 - b. Ein COS KANN weitere Werte für *algorithmIdentifier*
 - 1. unterstützen oder
 - 2. ablehnen.

(N018.000) K_COS

Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.11 beschriebenen Use Cases. Ist einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus der in (N017.900) genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:

- a. PSO Decipher (siehe 14.8.3),
- b. PSO Transcipher (siehe 14.8.6).

(N018.100) Dieser Punkt ist absichtlich leer.

(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)).

(N018.300) K_Anwendungsspezifikation {K_Karte}
Wertebereich von *algorithmIdentifier*

- a. Ist das Attribut *privateKey* des privaten Schlüsselobjektes vom Typ
 1. *privateRsaKey* MUSS zum Zwecke der Signaturerzeugung *algorithmIdentifier* aus folgender Menge gewählt werden (siehe Tabelle 270): {
 - i. sign9796_2_DS2, (siehe 14.8.2.2 und (N088.600)a)
 - ii. signPKCS1_V1_5, (siehe 14.8.2.1 und (N088.600)c)
 - iii. signPSS, (siehe 14.8.2.1 und (N088.600)c)
- b. Ein COS KANN weitere Werte für *algorithmIdentifier*
 1. unterstützen oder
 2. ablehnen.

(N018.400) K_COS

Bevor eines der folgenden Kommandos in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.9 beschriebenen Use Cases. Ist einem privaten Schlüsselobjekt ein *algorithmIdentifier* aus der in (N018.300) genannten Menge zugeordnet, dann MUSS es die folgenden Kommandos unterstützen:

- a. PSO Compute Digital Signature (siehe 14.8.2).

(N018.410) K_COS

Bevor das folgende Kommando in der Lage ist mit dem privaten Schlüsselobjekt zu arbeiten, ist es selektieren. Dies geschieht entweder mittels des in 14.9.9.9 beschriebenen Use Cases oder mittels eines Identifiers, der als Parameter dem zugreifenden Kommando mitgegeben wird. Ein privates Schlüsselobjekt MUSS folgendes Kommandos unterstützen:

- a. GENERATE ASYMMETRIC KEY PAIR (siehe 14.9.3).

(N018.420) K_COS

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. Neben den oben genannten Kommandos MUSS ein privates Schlüsselobjekt zusätzlich die folgenden administrativen Kommandos unterstützen:

- a. ACTIVATE (siehe 14.2.1),
- b. DEACTIVATE (siehe 14.2.3),

- c. DELETE (siehe 14.2.4).
- d. TERMINATE (siehe 14.2.9).

(N018.422) K_COS

Ein privates Schlüsselobjekt KANN weitere Kommandos

- a. unterstützen oder
- b. ablehnen.

Hinweis (40): 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)
- öffentliches Verschlüsselungsobjekte (siehe 8.6.4.3)

verwendet. Für öffentliche Schlüsselobjekte gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

(N018.500) K_Anwendungsspezifikation {K_Karte}

Ein öffentliches Schlüsselobjekt MUSS ein Attribut *keyIdentifier* besitzen.

(N018.550) K_Anwendungsspezifikation {K_Karte}

Eine öffentliches Schlüsselobjekt MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.

(N018.600) K_Anwendungsspezifikation {K_Karte}

Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut *publicKey* gemäß 8.2.4 besitzen.

(N018.700) K_Anwendungsspezifikation {K_Karte}

Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut vom Typ *oid* besitzen, welches angibt, für welche Zwecke es verwendet werden darf.

(N018.800) K_COS

Ein öffentliches Schlüsselobjekt MUSS genau ein Attribut *accessRules* vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen.

(N018.820) K_COS

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. Neben den in den folgenden Unterkapiteln genannten Kommandos MUSS ein öffentliches Schlüsselobjekt zusätzlich die folgenden administrativen Kommandos unterstützen:

- a. ACTIVATE (siehe 14.2.1),
- b. DEACTIVATE (siehe 14.2.3),
- c. DELETE (siehe 14.2.4).
- d. TERMINATE (siehe 14.2.9).

(N018.822) K_COS

Ein öffentliches Schlüsselobjekt KANN weitere Kommandos

- a. unterstützen oder
- b. ablehnen.

(N018.900) Diese Anforderung ist absichtlich leer.

Hinweis (41): 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:

(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.

(N019.100) K_Anwendungsspezifikation {K_Karte}

- a. Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge acht Oktett möglich sein.
- b. Ein COS KANN weitere Längen für *keyIdentifier*
 - 1. unterstützen oder
 - 2. ablehnen.

(N019.110) K_Anwendungsspezifikation {K_Karte}

Ein öffentliches Signaturprüfobjekt MUSS folgende Attribute vom Typ *interfaceDependentAccessRules* (siehe 8.1.4) besitzen, die an Schlüsselobjekte weitergereicht werden, die mittels CV-Zertifikat importiert werden (siehe (N095.900)):

- a. Genau ein Attribut *accessRulesPublicSignatureVerificationObject* und
- b. genau ein Attribut *accessRulesPublicAuthenticationObject*.

(N019.200) K_Anwendungsspezifikation {K_Karte}

Wertebereich des Attributes *oid*:

- a. Ist das Attribut *publicKey* des öffentlichen Signaturprüfobjektes vom Typ
 - 1. Option_RSA_CVC, *publicRsaKey* MUSS der Wert von *oid* Element der Menge {
 - i. sigS_ISO9796-2Withrsa_sha256,} sein (vergleiche Tabelle 271).
 - 2. *publicEkcKey* MUSS der Wert von *oid* Element der Menge {
 - i. ecdsa-with-SHA256,
 - ii. ecdsa-with-SHA384,
 - iii. ecdsa-with-SHA512} sein (vergleiche Tabelle 271) und zu den Domainparametern von *publicEkcKey* passen.

- b. Ein COS KANN weitere Werte für *oid*
 - 1. unterstützen oder
 - 2. ablehnen.

(N019.210) K_Anwendungsspezifikation {K_Karte}

Falls das Attribut *publicKey* vom Typ *publicElcKey* ist, dann gilt: Das öffentliche Signaturprüfobjekt MUSS ein Attribut

- a. CHAT enthalten, welches Aktionen kennzeichnet, die nach einer erfolgreichen Authentisierung mit einem öffentlichen Authentisierungsobjekt freigeschaltet werden.
- b. *expirationDate* vom Typ *date* (siehe (N008.120)) enthalten, welches angibt, ab welchem Zeitpunkt die Schlüsselverwendung endet.

(N019.300) K_COS

Bevor eines der folgenden Kommando in der Lage ist mit dem öffentlichen Signaturprüfobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.10 beschriebenen Use Cases. Ein öffentliches Signaturprüfobjekt MUSS zusätzlich zu den in (N018.820) genannten Kommandos die folgenden Kommandos unterstützen:

- a. PSO Verify Certificate (siehe 14.8.7)
- b. PSO Verify Digital Signature (siehe 14.8.9).

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:

(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.

(N019.500) K_Anwendungsspezifikation {K_Karte}

- a. Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge zwölf möglich sein.
- b. Ein COS KANN weitere Werte für *keyIdentifier*
 - 1. unterstützen oder
 - 2. ablehnen.

(N019.600) K_Anwendungsspezifikation {K_Karte}

Wertebereich des Attributes *oid*:

- a. Für öffentliche Authentisierungsobjekte des Typs
 - 1. Option_RSA_CVC, *publicRsaKey* MUSS der Wert von *oid* Element der Menge {
 - i. authS_ISO9796-2Withrsa_sha256_mutual,} sein (vergleiche Tabelle 271).
 - 2. *publicElcKey* MUSS der Wert von *oid* Element der Menge {
 - i. authS_gemSpec-COS-G2_ecc-with-sha256,}

- ii. `authS_gemSpec-COS-G2_ecc-with-sha384`,
 - iii. `authS_gemSpec-COS-G2_ecc-with-sha512`
- } sein (vergleiche Tabelle 271) und zu den Domainparametern von *publicElcKey* passen.
- b. Ein COS KANN weitere Werte für *oid*
 - 1. unterstützen oder
 - 2. ablehnen.
- (N019.700) K_Anwendungsspezifikation {K_Karte}
Falls *publicKey* vom Typ
- a. *Option_RSA_CVC*, *publicRsaKey* ist, MUSS das öffentliche Authentisierungsobjekt ein Attribut *accessRights* besitzen, welches eine *CHA* gemäß 7.1.1.4 speichert.
 - b. *publicElcKey* ist, 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 Schlüsselerwendung endet.
- (N019.800) K_COS
Bevor eines der folgenden Kommandos in der Lage ist mit dem öffentlichen Authentisierungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.5 beschriebenen Use Cases. Ein öffentliches Authentisierungsobjekt MUSS zusätzlich zu den in (N018.820) genannten Kommandos die folgenden Kommandos unterstützen:
- a. EXTERNAL AUTHENTICATE (siehe 14.7.1),
 - b. GENERAL AUTHENTICATE (siehe 15.4.4),
 - c. INTERNAL AUTHENTICATE (siehe (N086.900)d).

8.6.4.3 Öffentliches Verschlüsselungsobjekt

Ein öffentliches Verschlüsselungsobjekt wird zum Verschlüsseln von Daten eingesetzt. Für dieses Schlüsselobjekt gelten folgende Regeln, die bei der Spezifikation einer Anwendung einzuhalten sind:

- (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.
- (N019.822) K_Anwendungsspezifikation {K_Karte },
Option_Kryptobox
- a. Als Wert des Attributes *keyIdentifier* MUSS ein beliebiger Oktettstring der Länge zwölf möglich sein.
 - b. Ein COS KANN weitere Werte für *keyIdentifier*
 - 1. unterstützen oder
 - 2. ablehnen.

(N019.824) K_Anwendungsspezifikation {K_Karte},
Option_Kryptobox
Wertebereich des Attributes *oid*:

- a. Für öffentliche Verschlüsselungsobjekt des Typs
 1. *publicRsaKey* MUSS der Wert von *oid* Element der Menge {
 - i. id-RSAES-OAEP
 - ii. rsaEncryption} sein.
 2. *publicElcKey* MUSS der Wert von *oid* Element der Menge {
 - i. id-ELC-shared-secret-calculation} sein.
- b. Ein COS KANN weitere Werte für *oid*
 1. unterstützen oder
 2. ablehnen.

(N019.826) K_COS,
Option_Kryptobox
Ein öffentliches Verschlüsselungsobjekt MUSS das folgende Kommando unterstützen:
– PSO Encipher (siehe 14.8.4.3, 14.8.4.4 und 14.9.9.12),
Bevor dieses Kommando in der Lage ist mit dem öffentlichen Verschlüsselungsobjekt zu arbeiten, ist es zu selektieren. Dies geschieht mittels des in 14.9.9.12 beschriebenen Use Cases.

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.Verordnung) als auch allgemein bekannte Daten (EF.X509_Zertifikat) auslesbar (Read Binary).

- a. In der Umgebung_1 werden Lauschangriffe auf die physikalische Schnittstelle (siehe Abbildung 1) 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.
- b. In der Umgebung_2 existieren keine Angriffe auf die physikalische Schnittstelle (siehe Abbildung 1). Deshalb wird aus diversen Gründen die Zugriffsregel so gewählt, dass alle Daten ungeschützt übertragen werden.

Zu jeder Einsatzumgebung existiert ein Satz Zugriffsregeln:

- c. Umgebung_1:
 1. EF.Verordnung Read Binary an einem Kiosk mit Secure Messaging
 2. EF.X509_Zertifikat Read Binary immer erlaubt
- d. Umgebung_2:
 1. EF.Verordnung Read Binary in einer Apotheke ohne Verschlüsselung
 2. 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 14.9.9.1 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 datei-orientierten 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 (42): Die normativen Regeln dieses Dokumentes sind so gewählt, dass es für eine performante Implementierung zulässig ist, die kanalspezifischen Attribute (siehe 11.8) im RAM

zu speichern. Die genannten Attribute wurden lediglich der einfacheren Darstellung wegen, wie gezeigt zugeordnet.

Hinweis (43): 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 Bandbreite 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

- a. 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.
- b. 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 (44): 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 Ordnebenen 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. Falls die Suche dort erfolglos ist, 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:

(N019.900) K_Anwendungsspezifikation {K_Karte} und K_COS

Das COS MUSS ein hierarchisches Objektsystem mit mehreren Ebenen unterstützen.

- a. Das Objektsystem MUSS ein Attribut *root* mit Eigenschaften gemäß (N019.910) und (N019.920) besitzen:
- b. Das Objektsystem MUSS zwei Attribute "Answer To Reset" besitzen für die in (N024.100) Anforderungen genannt werden:

1. Das Attribut *coldAnswerToReset* vom Typ Oktettstring MUSS vom COS im Rahmen eines Cold Reset gemäß (N023.920)b versendet werden.
2. Das Attribut *warmAnswerToReset* vom Typ Oktettstring MUSS vom COS im Rahmen eines Warm Reset gemäß (N023.920)c versendet werden.
3. K_Anwendungsspezifikation MUSS identische Anforderungen an *coldAnswerToReset* und *warmAnswerToReset* enthalten.
4. Das COS KANN für *coldAnswerToReset* und *warmAnswerToReset* unterschiedliche Werte verwenden.
- c. Das Objektsystem MUSS ein Attribut *iccsn8* vom Typ Oktettstring besitzen. Für *iccsn8* MÜSSEN beliebige Werte mit acht Oktette möglich sein.
- d. Das Objektsystem MUSS ein Attribut *applicationPublicKeyList* unterstützen.
 1. Das COS MUSS eine beliebige Anzahl Listenelemente unterstützen.
 2. Als Listenelemente MUSS der Objekttyp „Öffentliches Schlüsselobjekt“ (siehe 8.6.4) unterstützt werden in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).
 3. Elemente der Liste *applicationPublicKeyList* MÜSSEN persistent gespeichert werden.
 4. Elemente der Liste *applicationPublicKeyList* KÖNNEN
 - i. zentral an einer Stelle oder
 - ii. dezentral über mehrere Ordner verteilt gespeichert werden.
- e. Das Objektsystem MUSS ein Attribut *persistentCache* unterstützen.
 1. Das COS MUSS eine beliebige Anzahl Listenelemente unterstützen.
 2. Die Anwendungsspezifikation SOLL eine Begrenzung der Anzahl der Listenelemente vorschreiben.
 3. Als Listenelemente MÜSSEN die Objekttypen „Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) und „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2) unterstützt werden in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).
 4. Elemente der Liste *persistentCache* MÜSSEN persistent gespeichert werden.
 5. Elemente der Liste *persistentCache* KÖNNEN
 - i. zentral an einer Stelle oder
 - ii. dezentral über mehrere Ordner verteilt gespeichert werden.
- f. Die Anwendungsspezifikation SOLL eine Begrenzung der Anzahl der Listenelemente in *persistentPublicKeyList* vorschreiben.
- g. Das Objektsystem MUSS ein Attribut *volatileCache* unterstützen.
 1. Listenlänge: Das COS KANN Listenlängen mit dem Wert null, eins oder größer unterstützen.
 2. Als Listenelemente MÜSSEN die Objekttypen „Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) und „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2) unterstützt werden in Verbindung mit einer Referenz zu einem Ordner, dem dieses Schlüsselobjekt zugeordnet ist (siehe (N021.600) und (N095.900)d).

- 3. Elemente der Liste *volatileCache* DÜRFEN NICHT persistent gespeichert werden.
- 4. Elemente der Liste *volatileCache* KÖNNEN
 - i. zentral an einer Stelle oder
 - ii. dezentral über mehrere Ordner verteilt gespeichert werden.
- h. Alle Elemente der Liste *allPublicKeyList* MÜSSEN in allen logischen Kanälen zur Verfügung stehen und shareable sein.
- i. Das Objektsystem MUSS genau ein Attribut vom Typ *lifeCycleStatus* (siehe 8.1.3) besitzen.
- j. Das Objektsystem MUSS genau ein Attribut *pointInTime* vom Typ *date* gemäß (N008.120) besitzen. *Hinweis: 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].*

(N019.910) K_Anwendungsspezifikation {K_Karte}

Für das Attribut *root* des Objektsystems gilt:

- a. Die Anwendungsspezifikation MUSS für *root* die Wahl zwischen den Typen Applikation (siehe 8.3.1.1) und Application Dedicated File (siehe 8.3.1.3) zulassen.
- b. Die Anwendungsspezifikation MUSS als Zugriffsbedingung für die Zugriffsart DELETE (siehe 14.2.4) NEVER (siehe (N022.100)) fordern.

(N019.920) K_COS

Für das Attribut *root* des Objektsystems gilt:

- a. Das COS MUSS für *root* den Typ Applikation (siehe 8.3.1.1) unterstützen.
- b. Das COS KANN für *root* weitere Ordnertypen unterstützen.
- c. Das COS KANN für *root* für die Zugriffsart DELETE beliebige Zugriffsbedingungen unterstützen.

(N019.930) K_Anwendungsspezifikation {K_Karte}

Im Objektsystem DARF es KEINE zwei verschiedenen Ordner geben, deren Attribut *applicationIdentifier* identisch ist.

(N020.000) K_COS

Der Ordner *root* wird der Ebene_0 zugeordnet. Ebene_0 wird als „höchste Ebene im Objektsystem“ bezeichnet.

(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) K_COS

Dieser Punkt ist absichtlich leer. In einer früheren Dokumentenversion enthielt er die Anforderung Applikationen entweder der Ebene_0 oder der Ebene_1 zuzuordnen, siehe dazu auch die Änderung in (N020.300).

(N020.300) K_COS

Ordner, die der Ebene_0 (*root*), Ebene_1 (DF2) oder Ebene_2 (DF3) 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.

(N020.390) K_Anwendungsspezifikation {K_Karte}

Ordner, die der Ebene_3 zugeordnet sind DÜRFEN KEIN Objekt des Typs Ordner enthalten.

(N020.400) K_COS

Ordner, die der Ebene_3 zugeordnet sind,

- a. KÖNNEN im Attribut *children* Ordner zulassen.
- b. KÖNNEN im Attribut *children* Ordner ablehnen.

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, 8.4, 8.5 und 8.6 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*:

(N020.500) K_COS

Als physikalischer Wert des *lifeCycleStatus* wird der Wert bezeichnet, den das entsprechende Attribut eines Objektes besitzt.

(N020.600) K_COS

Als logischer Wert des *lifeCycleStatus* wird der Wert bezeichnet, 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:

- a. „Operational state (active)“ < „Operational state (deactivated)“ < „Termination state“
- b. Für das Objekt *root* MUSS als logischer Wert der nächsthöheren Ebene der physikalische Wert des *lifeCycleStatus* des Objektsystems verwendet werden (siehe (N019.900)i).
- c. Für ein Objekt mit einem Attribut *lifeCycleStatus* MUSS gelten: Falls
 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 logische Wert des *lifeCycleStatus* des Objektes gleich seinem physikalischen.
- d. 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 Ordern 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 Passwortsuche

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 Abbildung 1) wird das im Folgenden festgelegte Verhalten deutlich:

Input:	<i>passwordReference</i>	Gemäß (N072.800)
	<i>startFolder</i>	Ordner, bei dem die Suche nach einem Passwortobjekt startet
Output:	<i>password</i>	Enthält das gefundene Passwortobjekt
Errors:	<i>pwdNotFound</i>	Zu den gegebenen Inputparametern wurde kein passendes Passwortobjekt gefunden
Notation:		<i>password</i> = SearchPwd(<i>startFolder</i> , <i>passwordReference</i>)

(N020.700) K_COS

Der Inputparameter *passwordReference* wird gemäß (N072.800) wie folgt zerlegt:

- a. *identifier* = *passwordReference* mod 128
- b. *location* = *passwordReference* – *identifier*

(N020.800) K_COS

Wenn *location* den Wert '00' = 0 besitzt und damit ein globales Passwort adressiert, dann wird im Wurzelverzeichnis *root* des Objektsystems nach einem Passwort gesucht, dessen Attribut *pwdIdentifier* identisch zu *identifier* ist. Wenn ein solches Passwortobjekt

- a. existiert, dann wird es als Outputparameter *password* verwendet.
- b. nicht existiert, wird der Fehler „*pwdNotFound*“ zurückgemeldet.

(N020.900) K_COS

Wenn *location* den Wert '80' = 128 besitzt und damit ein DF-spezifisches Passwort adressiert, dann wird die „lokale Variable“ *folder* auf *startFolder* gesetzt.

(N021.000) K_COS

Wenn *folder*

- a. auf das Wurzelverzeichnis *root* zeigt, dann wird der Fehler „*pwdNotFound*“ zurückgemeldet.
- b. ansonsten wird in *folder* nach einem Passwort gesucht, dessen Attribut *pwdIdentifier* identisch zu *identifier* ist. Wenn ein solches Passwortobjekt
 1. existiert, dann wird es als Outputparameter *password* verwendet.
 2. nicht existiert, wird *folder* auf den nächsthöheren Ordner gesetzt und der Algorithmus mit Schritt (N021.000)a fortgesetzt.

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.

Input:	<i>startFolder</i>	Ordner, aus dem heraus die Suche startet
	<i>identifier</i>	Für diesen Parameter sind die folgende Wertebereiche möglich: a) ein Oktett <i>keyReference</i> gemäß (N099.600), b) acht Oktett <i>keyIdentifier</i> gemäß (N019.100)a, c) zwölf Oktett <i>keyIdentifier</i> gemäß (N019.500)a, d) zwölf Oktett <i>keyIdentifier</i> gemäß (N019.822)a, e) herstellerspezifisch zur Auswahl von Sessionkeys
	<i>algID</i>	Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt, oder „Wildcard“
Output:	<i>key</i>	Enthält das gefundene Schlüsselobjekt
Errors:	<i>keyNotFound</i>	Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden
	<i>notSupported</i>	Der Schlüssel unterstützt den von <i>algID</i> geforderten Algorithmus nicht
Notation:		<i>key</i> = SearchKey(<i>startFolder</i> , <i>identifier</i> , <i>algID</i>)

(N021.050) K_COS

Enthält der Parameter *identifier* eine Schlüsselreferenz gemäß

- a. (N099.600) (d. h. symmetrisches Authentisierungsobjekt, symmetrisches Kartenverbindungsobjekt oder privates Schlüsselobjekt), dann gilt:
key = SearchSecretKey(*startFolder*, *identifier*, *algID*)
- b. (N019.100)a oder (N019.500)a oder (N019.822)a (d. h. ein öffentliches Schlüsselobjekt), dann gilt:
key = SearchPublicKey(*startFolder*, *identifier*, *algID*)
- c. 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, MÜSSEN sie auch gefunden werden.

9.2.3.1 Suche nach einem geheimen Schlüsselobjekt mittels *keyReference*

In diesem Kapitel werden symmetrische Authentisierungsobjekte gemäß 8.6.1 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 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 Abbildung 1) wird das im Folgenden festgelegte Verhalten deutlich:

Input:	<i>keyReference</i>	Gemäß (N099.600)
	<i>algID</i>	Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt, oder „Wildcard“
	<i>startFolder</i>	Ordner, bei dem die Suche nach einem Schlüsselobjekt startet
Output:	<i>key</i>	Enthält das gefundene Schlüsselobjekt
Errors:	<i>keyNotFound</i>	Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden
	<i>notSupported</i>	Der Schlüssel unterstützt den von <i>algID</i> geforderten Algorithmus nicht
Notation:		<i>key</i> = SearchSecretKey(<i>startFolder</i> , <i>keyReference</i> , <i>algID</i>)

(N021.100) K_COS

Der Inputparameter *keyReference* wird gemäß (N099.600) in die Bestandteile

- a. *identifizier* = *keyReference* mod 128
- b. *location* = *keyReference* – *identifizier*

(N021.200) K_COS

Wenn *location* den Wert '00' = 0 besitzt und damit ein globales Schlüsselobjekt adressiert, dann MUSS im Wurzelverzeichnis *root* des Objektsystems nach einem Schlüsselobjekt gesucht werden, dessen Attribut *keyIdentifier* identisch zu *identifizier* ist. Wenn ein solches Schlüsselobjekt

- a. existiert, dann wird es als Outputparameter *key* verwendet.
- b. nicht existiert, wird der Fehler „*keyNotFound*“ zurückgemeldet.

(N021.300) K_COS

Wenn *location* den Wert '80' = 128 besitzt und damit ein DF-spezifisches Schlüsselobjekt adressiert, dann MUSS die „lokale Variable“ *folder* auf *startFolder* gesetzt.

(N021.400) K_COS

Wenn *folder*

- a. auf das Wurzelverzeichnis *root* zeigt, dann wird der Fehler „*keyNotFound*“ zurückgemeldet.

- b. ansonsten wird in *folder* nach einem Schlüsselobjekt gesucht, dessen Attribut *keyIdentifier* identisch zu *identifier* ist. Wenn ein solches Schlüsselobjekt
 1. existiert, dann wird es als Outputparameter *key* verwendet.
 2. nicht existiert, wird *folder* auf den nächsthöheren Ordner gesetzt und der Algorithmus mit Schritt (N021.400)a fortgesetzt.

(N021.500) K_COS

Wenn der Outputparameter *key* ein

- a. symmetrischer Schlüssel ist und dessen Attribut *algorithmIdentifier* nicht zum Parameter *algID* passt, dann wird der Fehler „notSupported“ zurückgemeldet.
- b. privater Schlüssel ist, dann wird in den Elementen von *listAlgorithmIdentifier* nach einem Element gesucht, dessen *selIdentifier* (siehe (N017.400)b.1) identisch zum Attribut *selIdentifier* (siehe (N030.000)a) des Ordners ist, in welchem *key* als child enthalten ist. Es wird der Fehler "notSupported" zurückgemeldet, wenn ein solches Element
 1. nicht existiert, oder
 2. existiert, aber die Menge *setAlgorithmIdentifier* (siehe (N017.400)b.2) das Element *algID* nicht enthält.

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“ (s. Abbildung 1) wird das im Folgenden festgelegte Verhalten deutlich:

Input:	<i>identifier</i>	Gemäß (N018.500), (N019.100)a, (N019.500)a, (N019.822)a
	<i>algID</i>	Kryptographisches Verfahren, welches der zu suchende Schlüssel unterstützt
	<i>startFolder</i>	Ordner, aus dem heraus die Suche startet
Output:	<i>key</i>	Enthält das gefundene Schlüsselobjekt
Errors:	<i>keyNotFound</i>	Zu den gegebenen Inputparametern wurde kein passendes Schlüsselobjekt gefunden
	<i>notSupported</i>	Der Schlüssel unterstützt den von <i>algID</i> geforderten Algorithmus nicht
Notation		<i>key</i> = SearchPublicKey(<i>startFolder</i> , <i>identifier</i> , <i>algID</i>)

(N021.600) K_COS

Die „lokale Variable“ *folder* wird auf *startFolder* gesetzt. Dann werden folgende Schritte ausgeführt:

- a. Es wird in *allPublicKeyList* nach einem Schlüsselobjekt gesucht,
 1. dessen Attribut *keyIdentifier* identisch zu *identifier* ist und

2. das dem Ordner *folder* zugeordnet ist.
- b. Wenn ein solches Schlüsselobjekt
 1. existiert, dann wird es als Outputparameter *key* verwendet.
 2. nicht existiert und *folder* ist
 - i. gleich *root*, wird der Fehler „*keyNotFound*“ zurückgemeldet.
 - ii. ungleich *root*, dann wird *folder* auf den nächsthöheren Ordner gesetzt und der Algorithmus mit Schritt (N021.600)a fortgesetzt.

(N021.605) K_COS

Wenn das Attribut *oid* des Outputparameters *key* nicht zur vorgegebenen *algID* passt (siehe (N021.610)), genau dann MUSS der Fehler *notSupported* zurückgemeldet werden.

(N021.610) K_COS

Zum OID

- a. sigS_ISO9796-2Withrsa_sha256 (siehe Tabelle 271) MUSS genau *algID* gleich verifyCertificate (siehe Tabelle 270) passen.
- b. authS_ISO9796-2Withrsa_sha256_mutual (siehe Tabelle 271) MÜSSEN genau die *algID* aus folgender Menge passen (siehe Tabelle 268): {rsaRoleCheck, rsaSessionkey4SM, rsaSessionkey4TC}.
- c. ecdsa-with-SHA256, ecdsa-with-SHA384 und ecdsa-with-SHA512 (siehe Tabelle 271) MUSS genau *algID* gleich verifyCertificate (siehe Tabelle 270) passen.
- d. authS_gemSpec-COS-G2_ecc-with-sha256, authS_gemSpec-COS-G2_ecc-with-sha384 und authS_gemSpec-COS-G2_ecc-with-sha512 (siehe Tabelle 271) MÜSSEN genau die *algID* aus folgender Menge passen (siehe Tabelle 268): {elcAsynchronAdmin, elcRoleCheck, elcSessionkey4SM, elcSessionkey4TC}.
- e. id-RSAES-OAEP (siehe Tabelle 271) MUSS genau *algID* gleich rsaEncipherOaep (siehe Tabelle 269) passen.
- f. rsaEncryption (siehe Tabelle 271) MUSS genau *algID* gleich rsaEncipherPKCS1_V1_5 (siehe Tabelle 269) passen.
- g. id-ELC-shared-secret-calculation (siehe Tabelle 271) MUSS genau *algID* gleich elcSharedSecretCalculation (siehe Tabelle 269) passen.

9.3 Cache für öffentliche Schlüsselobjekte

Die in diesem Unterkapitel beschriebene Funktion speichert ein übergebenes „Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) oder „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2) im Cache.

Input:	<i>puk</i>	„Öffentliches Signaturprüfobjekt“ (siehe 8.6.4.1) oder „Öffentliches Authentisierungsobjekt“ (siehe 8.6.4.2)
	<i>folder</i>	Ordner, dem <i>puk</i> zuzuordnen ist
Output:	<i>persistent</i>	True => <i>puk</i> wurde in <i>persistentCache</i> gespeichert False => <i>puk</i> wurde in <i>volatileCache</i> gespeichert
	Notation	<i>persistent</i> = StoreInCache(<i>folder</i> , <i>puk</i>)

(N021.630) K_COS, Option_RSA_CVC

Wenn *puk.publicKey* vom Typ *publicRsaKey* ist, dann gilt folgendes:

- a. *puk* MUSS in Cache gespeichert werden.
- b. *puk* KANN in *persistentCache* gespeichert.
- c. *puk* KANN in *volatileCache* gespeichert.

(N021.633) K_COS

Wenn *puk.publicKey* vom Typ *publicElcKey* ist, dann gilt folgendes:

- a. *puk* MUSS in Cache gespeichert werden.
- b. Wenn in Cache ein Schlüsselobjekt mit identischem Attribut *keyIdentifier* enthalten ist und Schlüssellänge und Schlüsseltyp (RSA oder ELC)
 1. stimmen nicht überein, dann KANN der COS-Hersteller das weitere Verhalten des COS festlegen.
 2. stimmen überein, dann wird das im Cache vorhandene Schlüsselobjekt mit identischem *keyIdentifier* durch *puk* ersetzt.
- c. Wenn in Cache kein Schlüsselobjekt mit identischem Attribut *keyIdentifier* enthalten ist, gilt folgendes:
 1. Wenn in *persistentCache* nicht genügend Platz vorhanden ist, dann MUSS gemäß (N021.636) freier Platz geschaffen werden.
 2. Wenn in *persistentCache*
 - i. genügend Platz vorhanden, dann MUSS *puk* in *persistentCache* gespeichert werden.
 - ii. nicht genügend Platz vorhanden, dann MUSS *puk* in *volatileCache* gespeichert werden.

(N021.636) K_COS

Freien Platz in Cache 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
 1. existieren, dann wird eines oder mehrere aus *persistentCache* entfernt. Wenn dadurch genügend Platz geschaffen wird, dann bricht dieser Algorithmus ab, sonst wird dieser Schritt wiederholt.
 2. 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. Falls solche Schlüsselobjekte
 1. existieren, dann wird eines aus *persistentCache* entfernt. Wenn dadurch genügend Platz geschaffen wird, bricht dieser Algorithmus ab, sonst wird dieser Schritt wiederholt.
 2. 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*
 - 1. genügend freier Platz vorhanden ist, dann bricht dieser Algorithmus ab.
 - 2. nicht genügend Platz vorhanden, dann MUSS ein beliebiges Schlüsselobjekt aus Cache entfernt und dieser Schritt wiederholt werden.

(N021.638) K_COS

Wird ein öffentliches Schlüsselobjekt aus Cache entfernt, so

- a. DARF dies in KEINEM logischen Kanal Auswirkungen auf *globalSecurityList* (siehe (N029.900)e) haben.
- b. DARF dies in KEINEM logischen Kanal Auswirkungen auf *dfSpecificSecurityList* (siehe (N029.900)f) haben.
- c. DARF dies in KEINEM logischen Kanal Auswirkungen auf *bitSecurityList* (siehe (N029.900)h) haben.
- d. KÖNNEN in beliebigen logischen Kanälen Einträge in *keyReferenceList*, welche eine Referenz auf das entfernte Schlüsselobjekt enthalten
 - 1. verändert werden, oder
 - 2. unverändert bleiben.

(N021.639) K_COS

Für den Rückgabewert gilt: Wenn *puk* in *persistentCache* gespeichert wurde, dann ist *persistent* = True, andernfalls False.

Hinweis (45): 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 (46): In (N021.636) ist eine Strategie beschrieben, die in mehreren Schritten versucht ein „unwichtiges“ Schlüsselobjekt im Cache zu finden. Falls so ein „unwichtiges“ Schlüsselobjekt existiert, 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 (47): 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 (48): 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 (49): 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:

(N021.700) K_Anwendungsspezifikation {K_Karte}

Eine Zugriffsart MUSS eine Liste von Kommandobeschreibungen sein.

(N021.800) K_Anwendungsspezifikation {K_Karte}

Als Untermenge zu [ISO/IEC 7816-4#Tabelle 32] MUSS für jedes Listenelement gelten:

- a. 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). Dabei gilt:
 1. Das CLA Byte in einer Zugriffsart MUSS die Kanalnummer null enthalten.
 2. Das CLA Byte in einer Zugriffsart DARF KEIN Secure Messaging anzeigen.
- b. Innerhalb eines Listenelementes ist der Parameter P1 (siehe 11.5.3) optional. Das bedeutet: Es MUSS möglich sein, dass ein Listenelement
 1. genau einen Parameter P1 enthält.
 2. keinen Parameter P1 enthält.
- c. Innerhalb eines Listenelementes ist der Parameter P2 (siehe 11.5.4) optional. Das bedeutet: Es MUSS möglich sein, dass ein Listenelement
 1. genau einen Parameter P2 enthält.
 2. keinen Parameter P2 enthält.
- d. Ein COS KANN weitere Kommandobeschreibungen
 1. unterstützen oder
 2. ablehnen.

(N021.900) K_COS

Definition: Bei der Auswertung der Zugriffsart MUSS diese genau dann zur aktuellen Kommando-APDU passen, wenn die Bestandteile der Zugriffsart identisch sind zu den Bestandteilen der Kommando-APDU an der Schnittstelle „Interpreter“ (siehe Abbildung 1).

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:

objectFolder Falls 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*

(N022.000) K_COS

Das boolesche Element ALWAYS MUSS stets den Wert True liefern.

(N022.100) K_COS

Das boolesche Element NEVER MUSS stets den Wert False liefern.

(N022.200) K_COS

Das boolesche Element PWD(*passwordReference*) MUSS

- a. genau dann True liefern, wenn die Passwortsuche gemäß 9.2.2 mit den Inputparametern SearchPwd(*objectFolder*, *passwordReference*) ein Passwort findet und wenigstens eine der beiden folgenden Bedingungen erfüllt ist:

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. Dabei MUSS das Attribut *securityStatusEvaluationCounter* dieses Passwortes um eins dekrementiert werden. Falls durch das Dekrementieren der Wert auf Null gefallen ist,

- i. KANN der Eintrag in der Liste verbleiben oder
 - ii. KANN der Eintrag aus der Liste entfernt werden.

2. Das Attribut *flagEnabled* dieses Passwortes besitzt den Wert False.

- b. in allen anderen Fällen False liefern.

(N022.300) K_COS

Das boolesche Element AUT(*keyReference*) MUSS

- a. genau dann True liefern, wenn die Schlüsselsuche gemäß 9.2.3.1 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.

- b. in allen anderen Fällen False liefern.

(N022.400) K_COS

Das boolesche Element AUT(CHAT) MUSS

- a. genau dann True liefern, 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 Hinweis (51):) und dieses Element ist einem Ordner in *path(objectFolder)* zugeordnet (siehe (N030.400)b).
- b. in allen anderen Fällen False liefern.

(N022.500) K_COS, Option_RSA_CVC

Das boolesche Element AUT(CHA) MUSS

- a. genau dann True liefern, 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).
- b. in allen anderen Fällen False liefern.

(N022.600) K_COS

Secure Messaging gemäß Kapitel 13 wird wie folgt in Zugriffsbedingungen verwendet:

- a. Das boolesche Element *SmMac(keyInformation)* MUSS genau dann den Wert True liefern, 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
 - i. *Option_RSA_CVC*, *CHA* gemäß 7.1.1.4 und *AUT(keyInformation)* gemäß (N022.500) liefert True, oder
 - ii. *CHAT* und *AUT(keyInformation)* gemäß (N022.400) liefert True,
 - 3. sonst False.
- b. Das boolesche Element *SmCmdEnc* MUSS den Wert von *SessionkeyContext.flagCmdEnc* liefern.
- c. Das boolesche Element *SmRspEnc* MUSS stets den Wert True liefern.
- d. *SessionkeyContext.flagRspEnc* MUSS genau dann auf True gesetzt werden, wenn die Zugriffsbedingung das Element *SmRspEnc* enthält.

(N022.700) K_Anwendungsspezifikation {K_Karte}

Die zuvor definierten booleschen Elemente MÜSSEN sich zu einem booleschen Ausdruck verbinden lassen.

- a. Der boolesche Ausdruck MUSS den AND-Operator unterstützen.
- b. Der boolesche Ausdruck MUSS den OR-Operator unterstützen.
- c. Der boolesche Ausdruck DARF KEINEN anderen Operator enthalten.

(N022.710) K_Anwendungsspezifikation {K_Karte}

Eine Verschlüsselung ist nur bei gleichzeitiger MAC-Sicherung sinnvoll, das heißt:

- a. Jedes boolesche Element *SmCmdEnc* MUSS durch den AND-Operator mit *SmMac(keyInformation)* verknüpft werden.

- b. Jedes boolesche Element SmRspEnc MUSS durch den AND-Operator mit SmMac(keyInformation) verknüpft werden.

(N022.800) K_COS

Der boolesche Wert der Zugriffsbedingung MUSS das Ergebnis des booleschen Ausdrucks sein.

(N022.810) K_COS

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 MUSS für die Zugriffsbedingung dieser Kommandos gelten: Falls ein COS für eines der genannten Kommandos

- a. keine Zugriffsregeln ausgewertet, dann MUSS sich das COS so verhalten, als ob die Zugriffsbedingung ALWAYS lautet.
- b. sonst gilt für die Zugriffsbedingung dieses Kommandos:
 1. Im Element eins von *interfaceDependentAccessRules* (kontaktbehaftete Kommunikation) MUSS die Zugriffsbedingung ALWAYS lauten.
 2. Im Element zwei von *interfaceDependentAccessRules* (kontaktlose Kommunikation) MUSS die Zugriffsbedingung
 - i. ALWAYS lauten, wenn beabsichtigt ist, diese Schnittstelle zu nutzen.
 - ii. NEVER lauten, wenn beabsichtigt ist, diese Schnittstelle nicht zu nutzen.

Hinweis (50): 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, s. (N015.800)c.2), oder
 - genau n Operationen möglich sind (startSSEC = $n > 1$), oder
 - beliebig viele Operationen möglich sind (startSSEC = „unendlich“).

Hinweis (51): 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 = {1000, 0111} ist AUT(1100) gleich False, weil es in bitSecurityList kein Element gibt, in welchem die beiden ersten Bits gesetzt sind.
- b. Für bitSecurityList = {1011, 1101} ist AUT(1100) gleich True, weil in bitSecurityList im zweiten Element ebenfalls die beiden ersten Bits gesetzt sind.

Hinweis (52): 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:

(N022.900) K_Anwendungsspezifikation {K_Karte}

Eine Zugriffsregel MUSS eine Liste mit mindestens einem Element sein.

(N023.000) K_Anwendungsspezifikation {K_Karte}

Ein Listenelement MUSS genau eine Zugriffsart gemäß 10.1 und eine Zugriffsbedingung gemäß 10.2 enthalten.

(N023.100) K_COS

Ein Listenelement MUSS

- a. genau dann den booleschen Wert True liefern, wenn
 1. die Zugriffsart gemäß (N021.900) passt und
 2. der boolesche Wert der Zugriffsbedingung den Wert True hat.
- b. in allen anderen Fällen False liefern (implizites NEVER).

(N023.200) K_COS

Eine Zugriffsregel MUSS

- a. genau dann als erfüllt gelten, wenn mindestens ein Listenelement den Wert True liefert.
- b. in allen anderen Fällen als nicht erfüllt gelten.

(N023.300) K_Anwendungsspezifikation {K_Karte}

Eine Zugriffsregel, welche die übrigen Bedingungen des Kapitels 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 ein bestimmtes Kommando erlaubt oder verboten ist.

Input:	<i>obj</i>	Instanz eines Objektes vom Typ Ordner, Datei, Passwortobjekt, symmetrisches Authentisierungsobjekt oder privates Schlüsselobjekt
	<i>CLA</i> <i>INS</i> <i>P1</i> <i>P2</i>	CLA Byte INS Byte Parameter P1 Parameter P2
Output:	<i>b</i>	Boolean, True bedeutet: Kommando erlaubt; False bedeutet: Kommando verboten
Errors:	–	Keine
Notation:		<i>b</i> = AccessRuleEvaluation(<i>obj</i> , <i>CLA</i> , <i>INS</i> , <i>P1</i> , <i>P2</i>)

(N023.400) K_COS

Schritt 1: Handelt es sich bei *obj*

- a. um einen Ordner, dann MUSS *se* = *seldentifier* dieses Ordners sein (siehe (N030.000)a).
- b. um einen Sessionkey, dann MUSS *se* = *seldentifier* des Ordners sein (siehe (N030.000)a), auf den *SessionkeyContext.folderSessionkeys* zeigt.

- c. sonst MUSS *se* = *seldentifier* des Ordners sein (siehe (N030.000)a), der *obj* in seinem Attribut *children* (siehe (N010.000)) enthält.

(N023.500) K_COS

Schritt 2: Unter Berücksichtigung des aktuell aktiven Übertragungsprotokolls (kontaktbehaftet oder kontaktlos) des logischen Wertes von *obj.lifeCycleStatus* (siehe (N020.600)) und *se* MUSS aus *obj.accessRules*, welches vom Typ *interfaceDependentAccessRules* ist, die passende Zugriffsregel ausgewählt werden (siehe 8.1.4).

(N023.600) K_COS

Schritt 3: Anhand von *CLA*, *INS*, *P1* und *P2* MUSS ermittelt werden, ob die in (N023.500) ausgewählte Zugriffsregel erfüllt ist (siehe 10.3).

(N023.700) K_COS

Schritt 4: Ist die ausgewählte Zugriffsregel

- a. erfüllt, dann MUSS *b* = True gelten.
- b. sonst MUSS *b* = False gelten.

Hinweis (53): Falls in folgender Zugriffsbedingung PWD1 OR PWD2 beide Passwortobjekte mit SSEC ungleich unendlich vorkommen, dann ist es herstellerspezifisch, ob beide SSEC oder nur einer und in diesem Falle welcher dekrementiert wird.

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 Anforderungen:

Hinweis (54): 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.

(N023.740) K_COS

Auslesen von Zugriffsregeln:

- a. Jede Zugriffsregel, die in einer Anwendungsspezifikation definiert wird,
 - 1. MUSS auslesbar sein.
 - 2. SOLL frei auslesbar sein, das heißt die Zugriffsregel zum Auslesen von Zugriffsregeln SOLL „Read Always“ sein.
- b. Zugriffsregeln, die nicht in einer Anwendungsspezifikation definiert werden (etwa für herstellerspezifische zusätzliche Objekte EF.Pwd, etc.)
 - 1. KÖNNEN auslesbar sein,
 - 2. KÖNNEN nicht auslesbar sein.

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 kontaklos 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.

(N023.801) K_COS

Das COS MUSS das kontaktbehaftete Übertragungsprotokoll T=1 unterstützen. Das Intervall [(N023.820), (N024.220)] spezifiziert die diesbezüglichen Anforderungen.

(N023.820) K_IC

Elektrische Eigenschaften:

- a. Das IC einer Smartcard MUSS elektrische Eigenschaften gemäß [EMV®_Book-1#5.3] aufweisen.
- b. Das IC einer Smartcard MUSS die Spannungsstufe „Class A“ unterstützen.
- c. Das IC einer Smartcard MUSS die Spannungsstufe „Class B“ unterstützen.
- d. Das IC einer Smartcard KANN die Spannungsstufe „Class C“ unterstützen.
- e. Das IC einer Smartcard KANN die Unterstützung der Spannungsstufe „Class C“ ablehnen.
- f. Alle Spannungsstufen, die das IC einer Smartcard im ATR oder in einem EF.ATR anzeigt, MÜSSEN unterstützt werden und konform zu [EMV®_Book-1#5.3] implementiert sein (siehe auch (N024.100)i).

(N023.900) Diese Anforderung ist absichtlich leer.

(N023.920) K_IC, K_externeWelt {K_Karte}

Eine Kartensession gliedert sich in die Abschnitte Kartenaktivierung, Reset, Nachrichtenaustausch und Deaktivierung. Dabei gilt:

- a. Der externe Kommunikationspartner MUSS die Smartcard gemäß [EMV®_Book-1#6.1.2] aktivieren.
- b. 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.
- c. Das IC einer Smartcard MUSS einen Warm Reset gemäß [EMV®_Book-1#6.1.3.2] unterstützen.
- d. Die „Information Field Size“ IFSD MUSS durch einen IFSD-Request (siehe [ISO/IEC 7816-3#11.4.2]) auf den Wert 'FE' = 254 gesetzt werden.
- e. Der externe Kommunikationspartner KANN nach Abschluss eines Resets mittels des hier spezifizierten Übertragungsprotokolls beliebige Nachrichten austauschen.
- f. Deaktivierung:
 1. Der externe Kommunikationspartner SOLL das IC einer Smartcard gemäß [EMV®_Book-1#6.1.5] deaktivieren.
 2. Die Deaktivierung KANN auf andere Art erfolgen.
 3. 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 14.1.1 bzw. 14.1.2 verfahren wird, bevor die nächste Kommando-APDU von der Komponente „Cmd Interpreter“ in Abbildung 1 bearbeitet wird.

(N023.940) K_COS

Falls 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.

(N024.020) K_IC, K_externeWelt {K_Karte}

Im Rahmen des hier spezifizierten Übertragungsprotokolls MÜSSEN auf der physikalischen Schnittstelle (siehe Abbildung 1) Character gemäß [EMV®_Book-1#7.2] verwendet werden.

(N024.100) K_COS

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:

- a. Das COS MUSS den ATR gemäß [EMV®_Book-1#8.1] versenden.
- b. 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.
- c. Das COS MUSS im Rahmen eines Cold Reset für das TA1 Byte im ATR einen Wert aus der Menge {'18', '95', '96', '97'} verwenden.
- d. Das COS MUSS im Rahmen eines Warm Reset für das TA1 Byte im ATR einen Wert aus der Menge {'18', '95', '96'} verwenden.

- e. Für das TC1 Byte im ATR gilt:
 - 1. Das COS SOLL im ATR ein TC1 Byte enthalten.
 - 2. Das COS KANN im ATR auf ein TC1 Byte verzichten.
 - 3. Falls das TC1 Byte im ATR vorhanden ist, dann MUSS es den Wert 'FF' besitzen.
- f. Das COS MUSS im TD1 Byte des ATR das Protokoll T=1 anzeigen.
- g. Das COS DARF in den ATR kein TA2 Byte einstellen. Daraus folgt, dass das COS den „negotiable mode“ unterstützen MUSS (siehe (N024.220)).
- h. 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.
- i. 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.
- j. Falls 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.

(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:

- a. Falls TA1 im ATR den Wert '18' besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {'12', '13', '18'} akzeptieren.
- b. Falls TA1 im ATR den Wert '95' besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {'92', '93', '94', '95'} akzeptieren.
- c. Falls TA1 im ATR den Wert '96' besitzt, dann MUSS das COS in PPS1 einen Wert aus der Menge {'92', '93', '94', '95', '96'} akzeptieren.
- d. Falls 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.

(N024.320) K_COS, K_externeWelt {K_Karte}

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.

(N024.810) K_IC, Option_USB_Schnittstelle

Das IC einer Smartcard MUSS das kontaktbehaftete Übertragungsprotokoll gemäß [ISO/IEC 7816-12] unterstützen, wobei gilt:

- a. Bulk Transfer gemäß [ISO/IEC 7816-12#8.1] MUSS unterstützt werden.
- b. Weitere Übertragungsarten aus [ISO/IEC 7816-12]
 1. KÖNNEN unterstützt werden,
 2. KÖNNEN fehlen.

(N024.820) K_IC, Option_USB_Schnittstelle

Für die Übertragungsgeschwindigkeit des hier betrachteten Protokolls gilt:

- a. Das IC MUSS die Übertragungsgeschwindigkeit „Low Speed“ unterstützen.
- b. Das IC KANN weitere Übertragungsgeschwindigkeiten
 1. anbieten oder
 2. ablehnen.

(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.

(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:

- a. Das IC und das COS MÜSSEN einen Typ aus der Menge {„Typ A“, „Typ B“} unterstützen.
- b. Das IC und das COS
 1. MÜSSEN alle Datenraten aus der Menge $\{f_C / 128, f_C / 64, f_C / 32, f_C / 16\}$ unterstützen und
 2. KÖNNEN weitere Datenraten unterstützen.
- c. Die gesamte Karte bestehend aus IC, COS und Antenne im Kartenkörper

1. MUSS alle Datenraten aus der Menge $\{f_C / 128\}$ unterstützen.
2. SOLL alle Datenraten aus der Menge $\{f_C / 64, f_C / 32, f_C / 16\}$ unterstützen.
3. KANN weitere Datenraten unterstützen.

(N025.320) K_IC, Option_kontaktlose_Schnittstelle

Das IC einer Smartcard MUSS 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 Abbildung 1 wird exemplarisch gezeigt, wie eine Kommando-APDU CmdApu1 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 CmdApu1 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.

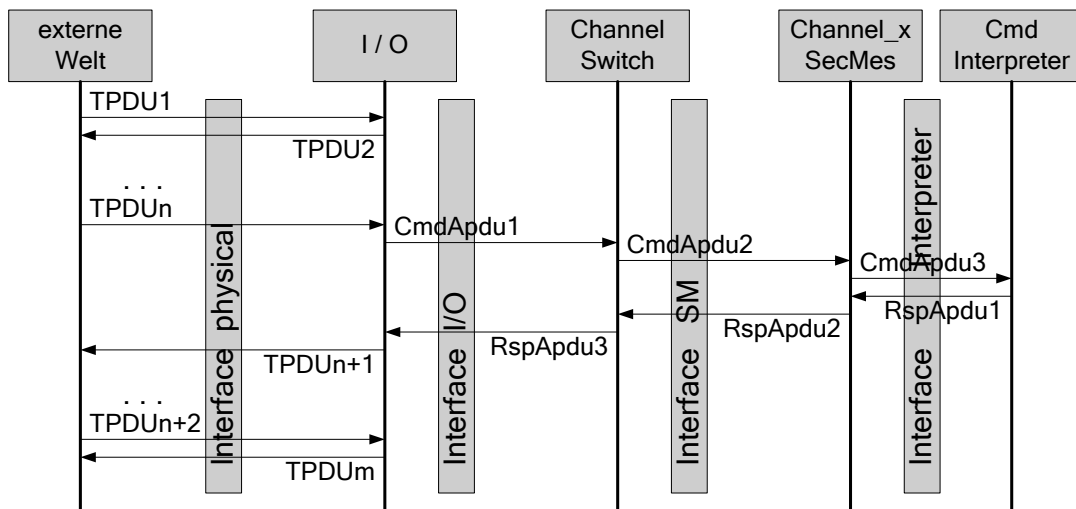


Abbildung 1: Message Sequence Chart für die Kommandobearbeitung

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:

(N025.600) K_externeWelt {K_Karte}

Ein externer Kommunikationspartner sendet über eine physikalische Schnittstelle eine Kommando-APDU CmdApu1, welche 11.5 entsprechen MUSS. Dazu wird CmdApu1 auf der physikalischen Schnittstelle entsprechend dem Übertragungsprotokoll (siehe 11.2) als (eine oder mehrere) TPDU transportiert.

(N025.650) K_COS

Das I/O der Smartcard MUSS die TPDU zusammensetzen und gewinnt dabei CmdApu1 zurück.

(N025.700) K_COS, K_externeWelt {K_Karte}

Die Fehlerbehandlung auf der physikalischen Schnittstelle MUSS [EMV®_Book-1] entsprechen.

(N025.800) K_COS

Anschließend MUSS die Kanalnummer aus dem CLA Byte von CmdApu1 extrahiert werden.

(N025.900) K_COS

Falls der Kanal mit der Kanalnummer aus dem CLA Byte nicht geöffnet ist, MUSS

- die Bearbeitung des Kommandos terminieren.
- RspApu3 in diesem Fall keine Antwortdaten enthalten und besteht lediglich aus dem Trailer ChannelClosed = '68 81'.

(N026.000) K_COS

Entsprechend der Kanalnummer MUSS CmdApu1 unverändert als CmdApu2 an den entsprechenden logischen Kanal weitergeleitet werden.

(N026.100) Dieser Punkt ist absichtlich leer.

(N026.200) K_COS

Innerhalb des logischen Kanals MUSS CmdApu2 mit dem entsprechenden Kanal-kontext *channelContext* an den Secure Messaging Layer („SecMes“ in Abbildung 1) weitergeleitet und dort gemäß 13.1.2 bearbeitet werden. Das Ergebnis ist RspApu2.

(N026.300) K_COS

RspApu2 MUSS unverändert als RspApu3 an das I/O der Smartcard geschickt werden.

(N026.400) K_COS

Das I/O der Smartcard MUSS RspApu3 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.

(N026.500) K_externeWelt {K_Karte}

Gemäß [ISO/IEC 7816-3] MUSS das CLA Byte in einem Oktett codiert werden.

(N026.510) K_COS

Die Bits im CLA Byte MÜSSEN die Bedeutung gemäß Tabelle 9 oder Tabelle 10 haben.

Tabelle 9: 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 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 10: 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 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.

(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.

(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.

(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}.

(N026.900) K_COS

An der Schnittstelle „Interpreter“ (siehe Abbildung 1) gilt für Nc:

- a. Das COS MUSS für Nc alle Werte unterstützen, die sich aus (N029.892) ergeben.
- b. Ein COS KANN weitere Werte für Nc
 1. unterstützen oder
 2. ablehnen.
- c. Ein COS DARF eine APDU NICHT wegen eines Längenfehlers 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.

(N026.910) K_externeWelt {K_Karte}

Der Sender einer Kommando-APDU SOLL die Längenbeschränkung durch passende Wahl von Nc einhalten.

Hinweis (55): 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, mindestens 4096 = '1000' zu unterstützen.

Hinweis (56): 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:

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, ..., 65.535, WildCardShort, WildCardExtended}

(N027.000) K_COS

An der Schnittstelle „Interpreter“ (siehe Abbildung 1) gilt für Ne:

- a. Das COS MUSS für Ne alle Werte aus der Menge {1, 2, ..., 65.534, 65.535, WildCardShort, WildCardExtended} unterstützen.
- b. Ein COS DARF eine APDU NICHT wegen eines Längenfehlers abweisen, wenn die Längenbeschränkung aus (N027.000)a oder die gesamte Antwort-APDU die Längenbeschränkung aus EF.ATR erfüllt.

(N027.010) K_externeWelt {K_Karte}

Der Sender einer Kommando-APDU SOLL die Längenbeschränkung durch passende Wahl von Ne einhalten.

Hinweis (57): 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 (58): 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:

(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. 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 {1, ..., 65536}.

(N027.200) K_COS

An der Schnittstelle „Interpreter“ (siehe Abbildung 1) gilt für Nr: Nr MUSS kleiner gleich Ne der zugehörigen Kommando-APDU sein. Falls die zugehörige Kommando-APDU

a. kein LeFeld besitzt, dann MUSS Nr = 0 sein.

b. ein LeFeld besitzt, gilt für LeFeld gleich

1. '00' = WildCardShort: Nr MUSS kleiner gleich '100' = 256 sein.
2. '0000' = WildCardExtended: Nr MUSS kleiner gleich '10000' = 65.536 sein.
3. sonst: Nr MUSS kleiner gleich OS2I(LeFeld) sein.

11.6.2 Trailer

Der Trailer zeigt den Status nach Bearbeitung einer Kommandonachricht an. Tabelle 272 zeigt die in diesem Dokument definierten Trailer. Der Trailer enthält Statusangaben über die Bearbeitung des Kommandos. Dazu zählen auch Fehlerindikationen.

(N027.210) K_COS

Der Trailer (siehe [ISO/IEC 7816-4#Tabelle 1]) besteht aus zwei Oktetten.

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 11: 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:

(N027.300) K_externeWelt {K_Karte}

Eine Case 1 Kommando-APDU MUSS aus vier Oktetten bestehen.

(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 12: 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 13: 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:

(N027.500) K_externeWelt {K_Karte}

Eine Case 2 Short Kommando-APDU MUSS aus fünf Oktetten bestehen.

(N027.600) K_externeWelt {K_Karte}

Der Wert von Ne wird wie folgt in einem Oktett codiert, welches als LeFeld bezeichnet wird:

- Wenn Ne im Intervall [1, 255] liegt, dann MUSS diese ganze Zahl in einem Oktett codiert werden: $LeFeld = I2OS(Ne, 1)$.
- Wenn Ne den Wert WildCardShort besitzt, dann MUSS das LeFeld gleich '00' sein.

(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 14: 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:

(N027.800) K_externeWelt {K_Karte}

Eine Case 2 Extended Kommando-APDU MUSS aus sieben Oktetten bestehen.

(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.

(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 (59): 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 15: Case 2 Antwort-APDU

Inhalt	Beschreibung
Daten	Optionaler Bestandteil der Antwort-APDU. Falls vorhanden, dann gelten die Bestimmungen aus 11.6.1.
Trailer	Statusbytes SW1 und SW2

11.7.3 Case 3 Kommando

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 16: 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 der Menge [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:

(N028.100) K_externeWelt {K_Karte}

Eine Case 3 Short Kommando-APDU MUSS aus fünf Oktetten plus den Oktetten aus dem Datenfeld bestehen.

(N028.200) K_externeWelt {K_Karte}

Die Anzahl Nc der Oktette des Datenfeldes MUSS in einem Oktett codiert werden, welches als LcFeld bezeichnet wird: $LcFeld = I2OS(Nc, 1)$.

(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 17: 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 der Menge [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:

(N028.400) K_externeWelt {K_Karte}

Eine Case 3 Extended Kommando-APDU MUSS aus sieben Oktetten plus den Oktetten aus dem Datenfeld bestehen.

(N028.500) K_externeWelt {K_Karte}

Die Anzahl Nc der Oktette des Datenfeldes MUSS in zwei Oktette codiert werden, welche als LcFeld bezeichnet werden: $LcFeld = I2OS(Nc, 2)$.

(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 (60): 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 18: Case 3 Antwort-APDU

Inhalt	Beschreibung
Trailer	Statusbytes SW1 und SW2

11.7.4 Case 4 Kommando

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 19: 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 der Menge [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:

(N028.700) K_externeWelt {K_Karte}

Eine Case 4 Short Kommando-APDU MUSS aus sechs Oktetten plus den Oktetten aus dem Datenfeld bestehen.

(N028.800) K_externeWelt {K_Karte}

Die Anzahl Nc der Oktette des Datenfeldes MUSS in einem Oktett codiert werden, welches als LcFeld bezeichnet wird: $LcFeld = I2OS(Nc, 1)$.

(N028.900) K_externeWelt {K_Karte}

Der Wert von Ne wird wie folgt in einem Oktett codiert, welches das LeFeld bildet:

- Wenn Ne im Intervall [1, 255] liegt, dann MUSS diese ganze Zahl als ein Oktett als LeFeld verwendet werden: $LeFeld = I2OS(Ne, 1)$.
- Wenn Ne den Wert WildCardShort besitzt, dann MUSS das LeFeld gleich '00' sein.

(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 20: 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 der Menge [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:

(N029.100) K_externeWelt {K_Karte}

Eine Case 4 Extended Kommando-APDU MUSS aus neun Oktetten plus den Oktetten aus dem Datenfeld bestehen.

(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).

(N029.300) K_externeWelt {K_Karte}

Der Wert von Ne wird wie folgt in zwei Oktette codiert, welche das LeFeld bilden:

- Wenn Ne im Intervall [1, 65535] liegt, dann MUSS diese ganze Zahl in zwei Oktette codiert werden: LeFeld = I2OS(Ne, 2).
- Wenn Ne den Wert WildCardExtended besitzt, dann MUSS das LeFeld gleich '0000' sein.

(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 (61): 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 21: Case 4 Antwort-APDU

Inhalt	Beschreibung
Daten	Optionalen Bestandteil der Antwort-APDU. Falls vorhanden, dann gelten die Bestimmungen aus 11.6.1.
Trailer	Statusbytes SW1 und SW2

(N029.500) Diese Anforderung ist absichtlich leer.

(N029.600) Diese Anforderung ist absichtlich leer.

(N029.610) Diese Anforderung ist absichtlich leer.

(N029.700) Diese Anforderung ist absichtlich leer.

(N029.800) Diese Anforderung ist 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 wurden, gilt für dieses Dokument folgendes Verhalten:

(N029.870) K_externeWelt {K_Karte}

Command Chaining ist eine konsekutive Folge von Kommando-Antwort-Paaren mit folgenden Eigenschaften:

- Mit Ausnahme des Bits b5 im CLA-Byte MÜSSEN alle Kommando-APDU einer Chaining-Kette identische Werte für CLA, INS, P1 und P2 haben.
- 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.
- Die Aussagen in (N029.870)a und (N029.870)b MÜSSEN für Kommando-APDU an der Schnittstelle "Interface I/O" aus Abbildung 1 gelten.

(N029.874) K_externeWelt {K_Karte}

Die konsekutive Folge einer Command-Chaining-Sequenz SOLL an der Schnittstelle "Interface I/O" aus Abbildung 1 NICHT

- durch Kommandos unterbrochen werden, welche nicht zu dieser Sequenz gehören oder
- durch eine Deaktivierung unterbrochen werden.

(N029.876) K_externeWelt {K_Karte}

Die konsekutive Folge einer Command-Chaining-Sequenz DARF an der Schnittstelle "Interface I/O" aus Abbildung 1 NICHT fortgesetzt werden, falls ein Kommando der Sequenz mit einem Fehler terminiert, das heißt, falls

- LOAD APPLICATION mit einem Trailer aus Tabelle 45 antwortet, oder
- GENERAL AUTHENTICATE mit einem Trailer aus Tabelle 174 antwortet.

(N029.878) K_COS

Falls die Anforderung (N029.874)a von der externen Entität nicht eingehalten wird, mithin also die Command-Chaining-Sequenz durch ein Kommando unterbrochen wird, dann

- a. MUSS das COS das unterbrechende Kommando akzeptieren.
- b. KANN das COS ein Fortsetzen der unterbrochenen Sequenz akzeptieren.
- c. SOLL das COS ein Fortsetzen der unterbrochenen Sequenz ablehnen.

(N029.880) K_COS

Falls 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

- a. SOLL das COS ein Fortsetzen der Sequenz ablehnen.
- b. KANN das COS ein Fortsetzen der Sequenz akzeptieren.

Hinweis (62): 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.

(N029.890) K_COS

Für die Längenbeschränkung von Kommando- und Antwortnachrichten gilt: Falls die Option_logische_Kanäle

- a. unterstützt wird, dann MUSS das COS folgende Intervalle unterstützen:
 - 1. ungeschützte Kommandonachricht: [4, 1.033] Oktett
 - 2. ungeschützte Antwortnachricht [2, 32.770] Oktett
 - 3. geschützte Kommandonachricht: [16, 1.033] Oktett
 - 4. geschützte Antwortnachricht [16, 1.033] Oktett
- b. nicht unterstützt wird, dann MUSS das COS folgende Intervalle unterstützen:
 - 1. ungeschützte Kommandonachricht: [4, 2.057] Oktett
 - 2. ungeschützte Antwortnachricht [2, 32.770] Oktett

- 3. geschützte Kommandonachricht: [16, 2.057] Oktett
- 4. geschützte Antwortnachricht [16, 2.057] Oktett
- c. Das COS KANN weitere Werte für die Längen von Nachrichten
 - 1. unterstützen oder
 - 2. ablehnen.

(N029.892) K_Anwendungsspezifikation {K_Karte}
Längenangaben im EF.ATR:

- a. Die Anwendungsspezifikation MUSS ein EF.ATR so spezifizieren, dass
 - 1. es als Kindobjekt im Ordner *root* (siehe (N019.900)a) enthalten ist und
 - 2. Längenbeschränkung sowohl für Kommando- und Antwortnachrichten als auch für geschützte und ungeschützte Nachrichten enthält, dabei gilt:
 - i. *limitCmdPlain* ist die maximale Anzahl von Oktetten in einer ungeschützt übertragenen Kommandonachricht.
 - ii. *limitCmdSecureMessaging* ist die maximale Anzahl von Oktetten in einer geschützt übertragenen Kommandonachricht.
 - iii. *limitRspPlain* ist die maximale Anzahl von Oktetten in einer ungeschützt übertragenen Antwortnachricht.
 - iv. *limitRspSecureMessaging* ist die maximale Anzahl von Oktetten in einer geschützt übertragenen Antwortnachricht.
- b. Die Angaben zu Längenbeschränkungen im EF.ATR MÜSSEN größer oder gleich zu den Längenbeschränkungen aus (N029.890) sein.

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) zugeordnet, oder durch das in 14.9.9.1 beschriebene Kommando *MANAGE SECURITY ENVIRONMENT* verändert.

12.1 Attribute eines logischen Kanals

An der physikalischen Schnittstelle (siehe Abbildung 1) verhält sich das COS so, als gäbe es für jeden logischen Kanal ein Objekt *channelContext* mit den folgenden Attributen:

(N029.900) K_COS

Folgende Attribute von *channelContext* sind dem Objektsystem zugeordnet: Der *channelContext* MUSS

- a. genau ein Attribut *currentFolder* enthalten, das auf ein Objekt vom Typ Ordner zeigt.
- b. 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)).
- c. 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:
 1. *externalAuthenticate*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
 2. *internalAuthenticate*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
 3. *verifyCertificate*, mit der Komponente *keyReference*.
 4. *signatureCreation*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
 5. *dataDecipher*, mit den Komponenten *keyReference* und *algorithmIdentifier*.
 6. *dataEncipher* mit den Komponenten *keyReference* und *algorithmIdentifier*.
 7. *macCalculation* mit den Komponenten *keyReference* und *algorithmIdentifier*.
- d. genau ein Attribut *SessionkeyContext* besitzen, welches folgende Elemente enthält:
 1. *flagSessionEnabled* für welches folgender Wertebereich definiert ist:

- i. noSK zeigt an, dass keine Sessionkeys vorhanden sind.
 - ii. SK4SM zeigt an, dass die übrigen Attribute gebrauchsfertiges kryptographisches Material enthalten, welches im Layer SecMes (siehe Abbildung 1) zum Entsichern einer Kommando-APDU oder zum Sichern einer Response-APDU verwendbar ist.
 - iii. SK4TC zeigt an, dass die übrigen Attribute gebrauchsfertiges kryptographisches Material enthalten, welches im Layer CmdInterpreter (siehe Abbildung 1) verwendbar ist (beispielsweise im Rahmen von PSO-Kommandos).
- 2. *Kenc* ist ein symmetrischer Schlüssel zur Ver- und Entschlüsselung.
- 3. *SSCenc* ist eine nicht-negative, ganze Zahl, die als Send Sequence Counter im Zusammenhang mit *Kenc* verwendet wird.
- 4. *Kmac* ist ein symmetrischer Schlüssel zur MAC-Berechnung und MAC-Verifikation.
- 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
 - i. 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).
 - ii. *accessRight* gemäß (N019.700) des beteiligten Schlüsselobjektes, wenn dieses folgenden Typ besitzt: öffentliches Authentisierungsobjekt (siehe 8.6.4.2).
- 9. 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.
- e. genau eine Liste *globalSecurityList* besitzen.
 - 1. Das COS MUSS alle Werte des Intervalls [0, 3] für die Länge dieser Liste unterstützen.

2. Ein COS KANN längere Listen unterstützen.
- f. genau eine Liste *dfSpecificSecurityList* besitzen.
 1. Das COS MUSS alle Werte des Intervalls [0, 3] für die Länge dieser Liste unterstützen.
 2. Ein COS KANN längere Listen unterstützen.
- g. Jedes Element der Liste *globalSecurityList* und jedes Element der Liste *dfSpecificSecurityList* MUSS entweder
 1. 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), oder
 2. 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.
 3. 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.
- h. genau eine Liste *bitSecurityList* besitzen.
 1. Das COS MUSS alle Werte des Intervalls [0, 1] für die Länge dieser Liste unterstützen.
 2. Ein COS KANN längere Listen unterstützen.
 3. 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, (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.7)
- i. genau eine Liste *globalPasswordList* besitzen.
 1. Das COS MUSS alle Werte des Intervalls [0, 4] für die Länge dieser Liste unterstützen.
 2. Ein COS KANN längere Listen unterstützen.
- j. genau eine Liste *dfSpecificPasswordList* besitzen.
 1. Das COS MUSS alle Werte des Intervalls [0, 4] für die Länge dieser Liste unterstützen.
 2. Ein COS KANN längere Listen unterstützen.
- k. 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 14.6.6.1 mit diesem Passwortobjekt stattgefunden hat.
- l. Der Wertebereich von *securityStatusEvaluationCounter*
 1. MUSS alle Werte von *startSsec* (siehe (N015.800)c.2) umfassen und

2. KANN den Wert null beinhalten.
- m. genau ein Attribut *currentEF* enthalten, das
 1. entweder unbestimmt ist,
 2. oder auf ein Listenelement von *currentFolder.children* vom Typ Datei zeigt.

(N030.000) K_COS

Folgende Attribute von *channelContext* sind einem Ordner zugeordnet: Der *channelContext* MUSS für jeden Ordner im Objektsystem

- a. genau ein Attribut *seldentifier* gemäß (N007.900) enthalten.

(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 gemäß 12.1 wird dann etabliert, wenn ein Reset durchgeführt wird, oder ein neuer logischer Kanal geöffnet wird. Dann gilt:

(N030.100) K_COS

Wenn der logische Kanal mit der Kanalnummer null (Basiskanal) durch

- a. Einschalten des physikalischen Interfaces, d. h.
 1. für das Übertragungsprotokoll T=1 gemäß [ISO/IEC 7816-3#11] wird eine Aktivierung gemäß (N023.920)a durchgeführt, oder
 2. für das USB-Übertragungsprotokoll gemäß 11.2.2 wird eine Aktivierung gemäß [ISO/IEC 7816-12] durchgeführt, oder
 3. für die kontaktlose Datenübertragung gemäß 11.2.3 wird eine Aktivierung gemäß ISO/IEC 14443 durchgeführt oder
 - b. durch einen Warm-Reset gemäß (N023.920)c
- geöffnet wird oder ein anderer logischer Kanal durch das Kommando MANAGE CHANNEL (siehe 14.9.8.1) geöffnet wird, oder ein logischer Kanal zurückgesetzt wird (siehe 14.9.8.3 und 14.9.8.4) dann MUSS für den betroffenen Kanal gelten:
- c. Dem betroffenen logischen Kanal MUSS exklusiv ein Kanalkontext gemäß 12.1 zugeordnet werden.
 - d. Das Attribut *currentFolder* MUSS auf *root* gesetzt werden.
 - e. Das Attribut *RND.ICC* MUSS auf den Wert "NoRandom" gesetzt werden.
 - f. Das Attribut *keyReferenceList* MUSS so gesetzt werden, dass alle Elemente leer sind.
 - g. Das Attribut *SessionkeyContext.flagSessionEnabled* MUSS auf noSK gesetzt werden.
 - h. Die Liste *globalSecurityList* MUSS leer sein.
 - i. Die Liste *dfSpecificSecurityList* MUSS leer sein.
 - j. Das Attribut *bitSecurityList* MUSS leer sein.
 - k. Die Liste *globalPasswordList* MUSS leer sein.

- l. Die Liste *dfSpecificPasswordList* MUSS leer sein.
- m. Für alle Ordner gilt:
 - 1. *selfIdentifier* MUSS auf den Wert eins gesetzt werden.
- n. Das Attribut *currentEF* MUSS auf den Wert „unbestimmt“ gesetzt werden.

Hinweis (63): 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.

Input:	<i>obj</i>	Ein Schlüsselobjekt
Output:	–	Kein Rückgabewert
Notation:		<code>setSecurityStatus(<i>obj</i>)</code>

(N030.200) K_COS

Falls *obj*

- a. im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalSecurityList* gelten (siehe (N029.900)e).
- b. in einem anderen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificSecurityList* gelten (siehe (N029.900)f).

(N030.300) K_COS

Falls *obj*

- a. ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) und *obj* in der Liste *tmpList*
 - 1. bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
 - 2. noch nicht vorhanden ist, dann MUSS *obj* am Anfang von *tmpList* eingetragen werden.
- b. *Option_RSA_CVC*, 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*
 - 1. bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
 - 2. 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.
- c. Falls *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).

(N030.400) K_COS

Falls *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)

- a. bereits vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
- b. 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.
- c. Falls *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.

Input:	<i>obj</i>	Ein Schlüsselobjekt
Output:	–	Kein Rückgabewert
Notation:		<i>clearSecurityStatusKey(obj)</i>

(N030.500) K_COS

Falls *obj*

- a. im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList = globalSecurityList* gelten (siehe (N029.900)e).
- b. in einem anderen Ordner zugeordnet ist, dann MUSS *tmpList = dfSpecificSecurityList* gelten (siehe (N029.900)f).

(N030.600) K_COS

Falls *obj*

- a. ein symmetrisches Authentisierungsobjekt (siehe 8.6.1) oder ein symmetrisches Kartenverbindungsobjekt (siehe 8.6.2) ist und *obj* in der Liste *tmpList*
 1. nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
 2. noch vorhanden ist, dann MUSS *obj* aus *tmpList* entfernt werden.
- b. Option_RSA_CVC, ein öffentliches Authentisierungsobjekt (siehe 8.6.4.2) ist und *obj.publicKey* ein
 1. RSA-Schlüssel ist und *obj.CHA* (siehe (N019.700)a) in der Liste *tmpList*
 - i. nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
 - ii. noch vorhanden ist, dann MUSS *obj.CHA* aus *tmpList* entfernt werden.

2. ELC-Schlüssel ist und *obj.CHAT* (siehe (N019.700)b.1) in der Liste *bitSecurityList* (siehe (N029.900)h).

- i. nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
- ii. noch vorhanden ist, dann MUSS *obj.CHAT* aus *bitSecurityList* entfernt werden.

c. einen anderen Typ besitzt, dann MUSS dieser Algorithmus beendet werden.

(N030.700) K_COS

Falls 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.

Input:	<i>obj</i>	Ein Ordner (siehe 8.3.1)
Output:	–	Kein Rückgabewert
Notation:		<code>clearSecurityStatusFolder(<i>obj</i>)</code>

(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.

Input:	–	Kein Parameter
Output:	–	Kein Rückgabewert
Notation:		<code>clearSessionkeys()</code>

(N030.740) K_COS

Der Sicherheitszustand, welcher bei der Aushandlung von Sessionkeys gesetzt wurde, ist wie folgt zu löschen:

- a. 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.

- b. *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.

Input:	<i>obj</i>	Ein Passwortobjekt
Output:	–	Kein Rückgabewert
Notation:		<code>setPasswordStatus(<i>obj</i>)</code>

(N030.800) K_COS

Falls *obj*

- im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalPasswordList* gelten (siehe (N029.900)i).
- einem anderen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificPasswordList* gelten (siehe (N029.900)j).

(N030.900) K_COS

Falls *obj* in der Liste *tmpList*

- bereits vorhanden ist, dann MUSS zunächst `clearPasswordStatus(obj)` und dann (N030.900)b ausgeführt werden.
- 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 *seldentifier* aus (N015.800)c.1 und *seldentifier* aus (N030.000)a ermittelt wird. Dabei MUSS in (N030.000) der Ordner zu Grunde gelegt werden, der *obj* enthält.

(N031.000) K_COS

Falls *tmpList* durch Eintragungen länger wurde, als vom COS unterstützt, dann MUSS das COS das letzte Listenelement entfernen (FIFO = first in first out).

(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.

Input:	<i>obj</i>	Ein Passwortobjekt
Output:	–	Kein Rückgabewert
Notation:		<code>clearPasswordStatus(<i>obj</i>)</code>

(N031.200) K_COS

Falls *obj*

- im Ordner *root* (siehe (N019.900)a) in der Liste *children* eingetragen ist, dann MUSS *tmpList* = *globalPasswordList* gelten (siehe (N029.900)i).
- in einem anderen Ordner zugeordnet ist, dann MUSS *tmpList* = *dfSpecificPasswordList* gelten (siehe (N029.900)j).

(N031.300) K_COS

Falls *obj* in der Liste *tmpList*

- nicht vorhanden ist, dann MUSS dieser Algorithmus beendet werden.
- 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 Abbildung 1. Dieser Layer benutzt neben den Informationen aus (N029.900)d folgende weitere Attribute:

- KD.i ist ein Oktettstring, der eine vom COS generierte Zufallszahl speichert, die im Rahmen der Ableitung von Sessionkeys verwendet wird.
- KD.e ist ein Oktettstring, der eine extern generierte Zufallszahl speichert, die im Rahmen der Ableitung von Sessionkeys verwendet wird.

13.1.1 Ableitung von Sessionkeys

Input:	–	Der benötigte Input wird dem <i>channelContext</i> entnommen
Output:	–	Kein Output vorhanden
Errors:	–	Keine
Notation:		SessionkeyDerivation()

Im Folgenden gelten die Definitionen:

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 (64): Bei der Aushandlung von Sessionkeys sind zwei Fälle zu unterscheiden: Entweder die Sessionkeyaushandlung erfolgte mittels

- a. symmetrischer Schlüssel, dann ist aus den obigen Definitionen SK relevant und PrK und PuK sind irrelevant, oder
- b. die Sessionkeyaushandlung erfolgte mittels asymmetrischer Schlüssel, dann ist aus den obigen Definitionen SK irrelevant und PrK und PuK sind relevant.

(N031.390) K_COS

Das Attribut *SessionkeyContext.negotiationKeyInformation* MUSS wie folgt gesetzt werden: Falls die Sessionkeys

a. asymmetrisch ausgehandelt wurden, dann gilt:
 $SessionkeyContext.negotiationKeyInformation = PuK.accessRights.$

b. symmetrisch ausgehandelt wurden, gilt:
 $SessionkeyContext.negotiationKeyInformation =$
 $channelContext.keyReferenceList.externalAuthenticate.keyReference.$

(N031.400) K_COS_G1, Option_DES

Falls der an der Aushandlung beteiligte Aushandlungsschlüssel die Aushandlung von 3DES-Schlüsseln impliziert, dann MUSS für die Attribute aus *SessionkeyContext* gelten:

a. $(Kenc, SSCenc, Kmac, SSCmac) = KeyDerivation_3DES(KD.i \text{ XOR } KD.e).$

b. Für *algIDSessionkey* in (N031.522) MUSS dann *algIDSessionkey* = desSessionkey gelten.

(N031.500) K_COS

Falls der an der Aushandlung beteiligte Aushandlungsschlüssel die Aushandlung von AES-Schlüsseln impliziert, dann MUSS für die Attribute aus *SessionkeyContext* gelten: Falls die Sessionkeys

a. symmetrisch ausgehandelt wurden und *SK* ein

1. AES-128 Schlüssel ist:

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES128(KD.i \text{ XOR } KD.e).$

2. AES-192 Schlüssel ist:

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES192(KD.i \text{ XOR } KD.e).$

3. AES-256 Schlüssel ist:

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES256(KD.i \text{ XOR } KD.e).$

b. asymmetrisch ausgehandelt wurden und *PrK.domainParameter* gleich

1. brainpoolP256r1 ist:

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES128(KD.i \text{ XOR } KD.e).$

2. brainpoolP384r1 ist:

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES192(KD.i \text{ XOR } KD.e).$

3. brainpoolP512r1 ist:

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES256(KD.i \text{ XOR } KD.e).$

c. mittels PACE ausgehandelt wurden und *channelContext.keyReferenceList.externalAuthenticate.algID* ist Element der Menge

1. {id-PACE-ECDH-GM-AES-CBC-CMAC-128, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128}

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES128(KD.i \text{ XOR } KD.e).$

2. {id-PACE-ECDH-GM-AES-CBC-CMAC-192, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192}

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES192(KD.i \text{ XOR } KD.e).$

3. {id-PACE-ECDH-GM-AES-CBC-CMAC-256, id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256}

$(Kenc, Kmac, SSCmac) = KeyDerivation_AES256(KD.i \text{ XOR } KD.e).$

d. Für *algIDSessionkey* in (N031.522) MUSS dann *algIDSessionkey* = aesSessionkey gelten.

(N031.520) K_COS

Falls *algId* anzeigt, dass die Sessionkeys im Rahmen von Secure Messaging einzusetzen sind, dann MUSS gesetzt werden

- a. *SessionkeyContext.flagSessionEnabled* = SK4SM und
- b. *SessionkeyContext.accessRulesSessionkeys* auf einen beliebigen Wert, weil die Zugriffsregeln der Sessionkeys im Rahmen dieser Spezifikation nicht ausgewertet werden (vergleiche auch (N031.522)b).

(N031.522) K_COS, Option_Kryptobox, Option_PACE_PCD

Falls *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:

- a. setze *SessionkeyContext.flagSessionEnabled* = SK4TC und
- b. setze *SessionkeyContext.accessRulesSessionkeys* gleich
 1. *SK.accessRulesSessionkeys* falls die Sessionkeys symmetrisch vereinbart wurden, oder
 2. *PrK.accessRulesSessionkeys*, falls die Sessionkeys asymmetrisch vereinbart wurden.
- c. setze *SessionkeyContext.folderSessionkeys* so, dass dadurch die Sessionkeys demselben Ordner zugeordnet werden wie *SK* bzw. *PrK*.
- d. trage in *channelContext.keyReferenceList.dataDecipher* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein und
- e. trage in *channelContext.keyReferenceList.dataEncipher* eine herstellerspezifische *keyReference* und *algorithmReference* = *algIDSessionkey* ein und
- f. 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.

(N031.600) K_COS

Falls im CLA-Byte kein Secure Messaging angezeigt wird (siehe [ISO/IEC 7816-4#5.4.1]) und *SessionkeyContext.flagSessionEnabled* den Wert

- a. SK4SM besitzt, dann
 1. MUSS der Sicherheitszustand des zugehörigen Aushandlungsschlüssels mittels *clearSessionkeys()* zurückgesetzt werden (siehe 12.4.3).
 2. MUSS *CmdApdu3* = *CmdApdu2* gesetzt werden.
- b. sonst, dann MUSS *CmdApdu3* = *CmdApdu2* gesetzt werden.

(N031.700) K_COS

Falls im CLA-Byte Secure Messaging angezeigt wird und *SessionkeyContext.flagSessionEnabled* besitzt den Wert

a. SK4SM, dann MUSS

1. die gesicherte CmdApdu2 den normativen Vorgaben aus 13.2 entsprechen. Falls ein DO mit Tag = '87' in der CmdApdu2 vorhanden ist, MUSS *flagCmdEnc* auf True gesetzt werden, sonst auf False. Das COS KANN weitere Secure Messaging Formate
 - i. akzeptieren oder
 - ii. ablehnen.
2. CmdApdu3 mittels der Umkehroperation zu den Regeln in 13.2 aus CmdApdu2 berechnet werden. Falls dabei ein Fehler festgestellt wird, genau dann
 - i. MUSS der Sicherheitszustand des zugehörigen Aushandlungsschlüssels mittels *clearSessionkeys()* zurückgesetzt werden (siehe 12.4.3).
 - ii. MUSS die Bearbeitung des Kommandos terminieren.
 - iii. RspApdu2 enthält in diesem Fall keine Antwortdaten und besteht lediglich aus dem Trailer *IncorrectSmDo* = '69 88'.

b. andernfalls

1. MUSS 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.

(N031.800) K_COS

Die vom Secure Messaging Layer („SecMes“ in Abbildung 1) gegebenenfalls weitergeleitete CmdApdu3 MUSS gemäß den Vorgaben aus Kapitel 14 bearbeitet werden, wobei eine Antwortnachricht-RspApdu1 entsteht.

Hinweis (65): 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.

(N031.900) K_COS

Falls *SessionkeyContext.flagSessionEnabled* den Wert

- a. SK4SM besitzt, dann MUSS die ungesicherte RspApdu1 gemäß 13.3 in die gesicherte RspApdu2 umgewandelt werden.
- b. sonst MUSS RspApdu2 = RspApdu1 gesetzt werden.

(N031.920) K_COS

Falls 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 13.1.1) und anschließend
- b. MÜSSEN KD.i und KD.e auf den Wert „nicht vorhanden“ gesetzt werden.

(N031.940) K_COS

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 Abbildung 1 wird nicht mit einer korrespondierenden Antwort-APDU RspApdu3 beantwortet. Gemäß den in Kapitel 11 beschriebenen Übertragungsprotokollen ist es dann nicht möglich, eine weitere Kommando-APDU zu schicken.

- a. In diesem Zustand KÖNNEN WTX-Requests über „Interface physical“ in Abbildung 1 gesendet werden, oder
- b. In diesem Zustand KANN ein Senden von WTX-Requests über „Interface physical“ in Abbildung 1 unterbleiben.
- c. Das COS MUSS diesen Zustand durch einen Reset verlassen.

Hinweis (66): Falls 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 Chiffre, 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

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
	<i>Kenc</i>	Symmetrischer Schlüssel für die Verschlüsselung, (N029.900)d.2
	<i>SSCenc</i>	Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der Verschlüsselung verwendet wird, (N029.900)d.3
	<i>Kmac</i>	Symmetrischer Schlüssel für die MAC-Berechnung, (N029.900)d.4
Output:	CmdApdu	Gesicherte Kommando-APDU
	Errors:	– Keine

Zur Sicherung einer generischen Kommando-APDU sind folgende Schritte durchzuführen:

(N032.000) K_externeWelt {K_Karte}

Falls das optionale Datenfeld Data fehlt, MUSS gelten:

ProtectedData = '' (leerer Oktettstring).

(N032.100) K_externeWelt {K_Karte}

Falls das optionale Datenfeld Data vorhanden ist und im Klartext übertragen wird, MUSS gelten:

a. Setze $\text{lenPDO} = \text{OctetLength}(\text{Data})$, falls lenPDO im Intervall

1. [1, 127] liegt, gilt: $\text{LenP} = \text{I2OS}(\text{lenPDO}, 1)$

2. [128, 255] liegt, gilt: $\text{LenP} = '81' \parallel \text{I2OS}(\text{lenPDO}, 1)$

3. [256, 65535] liegt, gilt: $\text{LenP} = '82' \parallel \text{I2OS}(\text{lenPDO}, 2)$

b. Setze $\text{ProtectedData} = '81' \parallel \text{LenP} \parallel \text{Data}'$

(N032.190) K_externeWelt {K_Karte}

SSCmac MUSS inkrementiert werden, das heißt: $\text{SSCmac} = \text{SSCmac} + 1$.

(N032.200) K_externeWelt {K_Karte}, Option_DES

Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird und *Kenc* ein 3TDES-Schlüssel ist, MUSS gelten:

a. $\text{SSCenc} = \text{SSCenc} + 1$

b. $C = \text{3TDES_CBC_ENC}(\text{Kenc}, \text{SSCenc}, \text{PaddingIso}(\text{Data}, 8))$

c. Setze $\text{lenCDO} = \text{OctetLength}(C) + 1$, falls lenCDO im Intervall

1. [1, 127] liegt, gilt: $\text{LenC} = \text{I2OS}(\text{lenCDO}, 1)$

2. [128, 255] liegt, gilt: $\text{LenC} = '81' \parallel \text{I2OS}(\text{lenCDO}, 1)$

3. [256, 65535] liegt, gilt: $\text{LenC} = '82' \parallel \text{I2OS}(\text{lenCDO}, 2)$

d. Setze $\text{ProtectedData} = '87' \parallel \text{LenC} \parallel 01 \parallel C'$

(N032.300) K_externeWelt {K_Karte}

Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird und *Kenc* ein AES-Schlüssel ist, MUSS gelten:

a. $\text{IVenc} = \text{OS2I}(\text{AES_ENC}(\text{Kenc}, \text{I2OS}(\text{SSCmac}, 16)))$

b. $C = \text{AES_CBC_ENC}(\text{Kenc}, \text{IVenc}, \text{PaddingIso}(\text{Data}, 16))$

c. Setze $\text{lenCDO} = \text{OctetLength}(C) + 1$, falls lenCDO im Intervall

1. [1, 127] liegt, gilt: $\text{LenC} = \text{I2OS}(\text{lenCDO}, 1)$

2. [128, 255] liegt, gilt: $\text{LenC} = '81' \parallel \text{I2OS}(\text{lenCDO}, 1)$

3. [256, 65535] liegt, gilt: $\text{LenC} = '82' \parallel \text{I2OS}(\text{lenCDO}, 2)$

d. Setze $\text{ProtectedData} = '87' \parallel \text{LenC} \parallel 01 \parallel C'$

(N032.400) K_externeWelt {K_Karte}

Für LeDO MUSS gelten: Falls das optionale LeFeld

a. fehlt, gilt: $\text{LeDO} = ''$, (leerer Oktettstring).

b. vorhanden ist, gilt: $\text{LeDO} = '97' \parallel \text{I2OS}(\text{OctetLength}(\text{LeFeld}), 1) \parallel \text{LeFeld}'$

(N032.500) K_externeWelt {K_Karte}

Wenn die Kanalnummer im CLA-Byte im Intervall

a. [0, 3] liegt, dann MUSS gesetzt werden $CLA' = CLA \text{ OR } '0C'$
Hinweis: Dadurch werden die Bits b4 und b3 in CLA gesetzt.

b. andernfalls MUSS gesetzt werden $CLA' = CLA \text{ OR } '20'$
Hinweis: Dadurch wird das Bit b6 in CLA gesetzt.

(N032.600) K_externeWelt {K_Karte}

Wenn die Kanalnummer im CLA-Byte im Intervall

a. [0, 3] liegt, dann MUSS gelten $head = CLA' \parallel INS \parallel P1 \parallel P2$

b. andernfalls MUSS gelten $head = '89\ 04' \parallel CLA' \parallel INS \parallel P1 \parallel P2$

(N032.700) Diese Anforderung ist absichtlich leer. Ihr Inhalt wurde nach (N032.190) verschoben.

(N032.800) K_externeWelt {K_Karte}

Es MUSS gelten: $tmpData = ProtectedData \parallel LeDO$

Falls *Kmac* ein

a. Option_DES, 3DES-Schlüssel ist, dann MUSS gelten:

1. $MACin = I2OS(SSCmac, 8)$

2. Falls die Kanalnummer im CLA-Byte größer gleich vier ist, oder $OctetLength(tmpData)$ gleich null ist

i. dann gilt: $MACin = MACin \parallel head \parallel tmpData$

ii. sonst: $MACin = MACin \parallel PaddingIso(head, 8) \parallel tmpData$

3. $MAC = CALCULATE_Retail_MAC(Kmac, MACin)$

b. AES-Schlüssel ist, dann MUSS gelten:

1. $MACin = I2OS(SSCmac, 16)$

2. Falls die Kanalnummer im CLA-Byte größer gleich vier ist, oder $OctetLength(tmpData)$ gleich null ist

i. dann gilt: $MACin = MACin \parallel head \parallel tmpData$

ii. sonst: $MACin = MACin \parallel PaddingIso(head, 16) \parallel tmpData$

3. $MAC = CalculateCMAC_IsoPadding(Kmac, MACin)$

(N032.900) K_externeWelt {K_Karte}

Für MDO MUSS gelten: $MDO = '8E \parallel I2OS(OctetLength(MAC), 1) \parallel MAC'$

(N033.000) K_externeWelt {K_Karte}

Für newD MUSS gelten: Wenn die Kanalnummer im CLA-Byte im Intervall

a. [0, 3] liegt, dann setze: $newD = tmpData \parallel MDO$

b. andernfalls setze: $newD = head \parallel tmpData \parallel MDO$

(N033.100) K_externeWelt {K_Karte}

Case 1: Falls Data und LeFeld fehlen, dann setze:

$CmdApdu = Case4S(CLA', INS, P1, P2, newD, WildCardShort)$

(N033.200) K_externeWelt {K_Karte}

Case 2: Falls Data fehlt und LeFeld vorhanden ist, dann MUSS gelten:

$CmdApdu = Case4E(CLA', INS, P1, P2, newD, WildCardExtended)$

(N033.300) K_externeWelt {K_Karte}

Case 3: Falls Data vorhanden ist und LeFeld fehlt und $OctetLength(newD)$

- a. kleiner gleich 255 ist, dann MUSS gelten:
CmdApdu = Case4S(CLA', INS, P1, P2, newD, WildCardShort)
- b. andernfalls MUSS gelten:
CmdApdu = Case4E(CLA', INS, P1, P2, newD, WildCardExtended)

(N033.400) K_externeWelt {K_Karte}

Case 4: Falls Data und LeFeld vorhanden sind, dann MUSS gelten:

CmdApdu = Case4E(CLA', INS, P1, P2, newD, WildCardExtended)

Hinweis (67): 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 (68): 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 gemäß 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 (Vorhandensein oder Fehlen von *flagRspEnc*, siehe (N022.600)d) entscheidet, ob vorhandene Responsedaten im Klartext oder als Chiffre 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:

Input:	Data	Optionaler Oktettstring gemäß 11.6.1
	Trailer	Oktettstring gemäß 11.6.2
	<i>Kenc</i>	Symmetrischer Schlüssel für die Verschlüsselung, (N029.900)d.2
	<i>SSCenc</i>	Beliebige nicht-negative Zahl, die als Send Sequence Counter bei der Verschlüsselung verwendet wird, (N029.900)d.3
	<i>Kmac</i>	Symmetrischer Schlüssel für die MAC-Berechnung, (N029.900)d.4

	<i>SSCmac</i>	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:

(N033.500) K_COS

Falls das optionale Datenfeld Data fehlt, MUSS gelten:
ProtectedData = '' (leerer Oktettstring).

(N033.600) K_COS

Falls das optionale Datenfeld Data vorhanden ist und im Klartext übertragen wird, MUSS gelten:

- a. Setze $\text{lenPDO} = \text{OctetLength}(\text{Data})$, falls lenPDO im Intervall
 1. [1, 127] liegt, gilt: $\text{LenP} = \text{I2OS}(\text{lenPDO}, 1)$
 2. [128, 255] liegt, gilt: $\text{LenP} = '81' \parallel \text{I2OS}(\text{lenPDO}, 1)$
 3. [256, 65535] liegt, gilt: $\text{LenP} = '82' \parallel \text{I2OS}(\text{lenPDO}, 2)$
- b. Setze ProtectedData = '81 || LenP || Data'

(N033.690) K_COS

SSCmac MUSS inkrementiert werden, das heißt: $\text{SSCmac} = \text{SSCmac} + 1$.

(N033.700) K_COS_G1, Option_DES

Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird (*flagRspEnc* ist True, siehe (N029.900)d.7 und (N022.600)d) und *Kenc* ein 3TDES-Schlüssel ist, MUSS gelten:

- a. $\text{SSCenc} = \text{SSCenc} + 1$
- b. $\text{C} = \text{3TDES_CBC_ENC}(\text{Kenc}, \text{SSCenc}, \text{PaddingIso}(\text{Data}, 8))$
- c. Setze $\text{lenCDO} = \text{OctetLength}(\text{C}) + 1$, falls lenCDO im Intervall
 1. [1, 127] liegt, gilt: $\text{LenC} = \text{I2OS}(\text{lenCDO}, 1)$
 2. [128, 255] liegt, gilt: $\text{LenC} = '81' \parallel \text{I2OS}(\text{lenCDO}, 1)$
 3. [256, 65535] liegt, gilt: $\text{LenC} = '82' \parallel \text{I2OS}(\text{lenCDO}, 2)$
- d. Setze ProtectedData = '87 || LenC || 01 || C'

(N033.800) K_COS

Falls das optionale Datenfeld Data vorhanden ist und verschlüsselt übertragen wird (*flagRspEnc* ist True, siehe (N029.900)d.7 und (N022.600)d) und *Kenc* ein AES-Schlüssel ist, MUSS gelten:

- a. $\text{IVenc} = \text{OS2I}(\text{AES_ENC}(\text{Kenc}, \text{I2OS}(\text{SSCmac}, 16)))$
- b. $\text{C} = \text{AES_CBC_ENC}(\text{Kenc}, \text{IVenc}, \text{PaddingIso}(\text{Data}, 16))$
- c. Setze $\text{lenCDO} = \text{OctetLength}(\text{C}) + 1$, falls lenCDO im Intervall
 1. [1, 127] liegt, gilt: $\text{LenC} = \text{I2OS}(\text{lenCDO}, 1)$
 2. [128, 255] liegt, gilt: $\text{LenC} = '81' \parallel \text{I2OS}(\text{lenCDO}, 1)$
 3. [256, 65535] liegt, gilt: $\text{LenC} = '82' \parallel \text{I2OS}(\text{lenCDO}, 2)$
- d. Setze ProtectedData = '87 || LenC || 01 || C'

(N033.900) K_COS

Für TDO MUSS gelten: $TDO = '99\ 02 \parallel \text{Trailer}'$

(N034.000) Diese Anforderung ist absichtlich leer. Ihr Inhalt wurde nach (N033.690) verschoben.

(N034.100) K_COS

Falls *Kmac* ein

a. Option_DES, 3TDES-Schlüssel ist, MUSS gelten:

1. $MAC_{in} = I2OS(SSC_{mac}, 8)$
2. $MAC_{in} = MAC_{in} \parallel ProtectedData$
3. $MAC_{in} = MAC_{in} \parallel TDO$
4. $MAC = CALCULATE_Retail_MAC(K_{mac}, MAC_{in})$

b. AES-Schlüssel ist, MUSS gelten:

1. $MAC_{in} = I2OS(SSC_{mac}, 16)$
2. $MAC_{in} = MAC_{in} \parallel ProtectedData$
3. $MAC_{in} = MAC_{in} \parallel TDO$
4. $MAC = CalculateCMAC_IsoPadding(K_{mac}, MAC_{in})$

(N034.200) K_COS

Für MDO MUSS gelten: $MDO = '8E \parallel I2OS(OctetLength(MAC), 1) \parallel MAC'$

(N034.300) K_COS

Für die gesicherte Antwort-APDU RspApdu MUSS gelten:

- a. $RspApdu.Datenfeld = ProtectedData \parallel TDO \parallel MDO$
- b. $RspApdu.Trailer = Trailer$

14 Kommandos (normativ)

Tabelle 22: Kommandos, alphabetisch

Kommando	CLA	INS	
ACTIVATE	'00'	'44'	14.2.1
ACTIVATE RECORD	'00'	'08'	14.4.1
APPEND RECORD	'00'	'E2'	14.4.2
CHANGE REFERENCE DATA	'00'	'24'	14.6.1
CREATE	'00'	'E0'	14.2.2
DEACTIVATE	'00'	'04'	14.2.3
DEACTIVATE RECORD	'00'	'06'	14.4.3
DELETE	'00'	'E4'	14.2.4
DELETE RECORD	'80'	'0C'	14.4.4
DISABLE VERIFICATION REQUIREMENT	'00'	'26'	14.6.2
ENABLE VERIFICATION REQUIREMENT	'00'	'28'	14.6.3
ENVELOPE	'00'	'C2'	14.9.1
ERASE BINARY	'00'	'0E'	14.3.1
ERASE RECORD	'00'	'0C'	14.4.5
EXTERNAL AUTHENTICATE	'00'	'82'	14.7.1
FINGERPRINT	'80'	'FA'	14.9.2
GENERAL AUTHENTICATE	'00'	'86'	14.7.2
GENERATE ASYMMETRIC KEY PAIR	'00'	'46'	14.9.3
GET CHALLENGE	'00'	'84'	14.9.4
GET DATA	'00'	'CA'	14.5.1
GET PIN STATUS	'80'	'20'	14.6.4
GET RANDOM	'80'	'84'	14.9.5
GET RESPONSE	'00'	'C0'	14.9.6
GET SECURITY STATUS KEY	'80'	'82'	14.7.3
INTERNAL AUTHENTICATE	'00'	'88'	14.7.4
LIST PUBLIC KEY	'80'	'CA'	14.9.7
LOAD APPLICATION	'00'	'EA'	14.2.5
MANAGE CHANNEL	'00'	'70'	14.9.8
MANAGE SECURITY ENVIRONMENT	'00'	'22'	14.9.9
MUTUAL AUTHENTICATE	'00'	'82'	14.7.1
PERFORM SECURITY OPERATION	'00'	'2A'	14.8
PUT DATA	'00'	'DA'	14.5.2
READ BINARY	'00'	'B0'	14.3.2
READ RECORD	'00'	'B2'	14.4.6
RESET RETRY COUNTER	'00'	'2C'	14.6.5
SEARCH BINARY	'00'	'A0'	14.3.3
SEARCH RECORD	'00'	'A2'	14.4.7
SELECT	'00'	'A4'	14.2.6
SET LOGICAL EOF	'80'	'0E'	14.3.4
TERMINATE	'00'	'E8'	14.2.9
TERMINATE CARD USAGE	'00'	'FE'	14.2.7
TERMINATE DF	'00'	'E6'	14.2.8
UPDATE BINARY	'00'	'D6'	14.3.5
UPDATE RECORD	'00'	'DC'	14.4.8
VERIFY	'00'	'20'	14.6.6
WRITE BINARY	'00'	'D0'	14.3.6
WRITE RECORD	'00'	'D2'	14.4.9

Tabelle 23: Kommandos, numerisch

Kommando	CLA	INS	
DEACTIVATE	'00'	'04'	14.2.3
DEACTIVATE RECORD	'00'	'06'	14.4.3
ACTIVATE RECORD	'00'	'08'	14.4.1
ERASE RECORD	'00'	'0C'	14.4.5
ERASE BINARY	'00'	'0E'	14.3.1
VERIFY	'00'	'20'	14.6.6
MANAGE SECURITY ENVIRONMENT	'00'	'22'	14.9.9
CHANGE REFERENCE DATA	'00'	'24'	14.6.1
DISABLE VERIFICATION REQUIREMENT	'00'	'26'	14.6.2
ENABLE VERIFICATION REQUIREMENT	'00'	'28'	14.6.3
PERFORM SECURITY OPERATION	'00'	'2A'	14.8
RESET RETRY COUNTER	'00'	'2C'	14.6.5
ACTIVATE	'00'	'44'	14.2.1
GENERATE ASYMMETRIC KEY PAIR	'00'	'46'	14.9.3
MANAGE CHANNEL	'00'	'70'	14.9.8
EXTERNAL AUTHENTICATE	'00'	'82'	14.7.1
MUTUAL AUTHENTICATE	'00'	'82'	14.7.1
GET CHALLENGE	'00'	'84'	14.9.4
GENERAL AUTHENTICATE	'00'	'86'	14.7.2
INTERNAL AUTHENTICATE	'00'	'88'	14.7.4
SEARCH BINARY	'00'	'A0'	14.3.3
SEARCH RECORD	'00'	'A2'	14.4.7
SELECT	'00'	'A4'	14.2.6
READ BINARY	'00'	'B0'	14.3.2
READ RECORD	'00'	'B2'	14.4.6
GET RESPONSE	'00'	'C0'	14.9.6
ENVELOPE	'00'	'C2'	14.9.1
GET DATA	'00'	'CA'	14.5.1
WRITE BINARY	'00'	'D0'	14.3.6
WRITE RECORD	'00'	'D2'	14.4.9
UPDATE BINARY	'00'	'D6'	14.3.5
PUT DATA	'00'	'DA'	14.5.2
UPDATE RECORD	'00'	'DC'	14.4.8
CREATE	'00'	'E0'	14.2.2
APPEND RECORD	'00'	'E2'	14.4.2
DELETE	'00'	'E4'	14.2.4
TERMINATE DF	'00'	'E6'	14.2.8
TERMINATE	'00'	'E8'	14.2.9
LOAD APPLICATION	'00'	'EA'	14.2.5
TERMINATE CARD USAGE	'00'	'FE'	14.2.7
DELETE RECORD	'80'	'0C'	14.4.4
SET LOGICAL EOF	'80'	'0E'	14.3.4
GET PIN STATUS	'80'	'20'	14.6.4
GET SECURITY STATUS KEY	'80'	'82'	14.7.3
GET RANDOM	'80'	'84'	14.9.5
LIST PUBLIC KEY	'80'	'CA'	14.9.7
FINGERPRINT	'80'	'FA'	14.9.2

(N034.400) K_COS

Das COS KANN Kommando-APDU unterstützen, die in diesem Kapitel 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 undefinierten Zustand umzugehen ist.

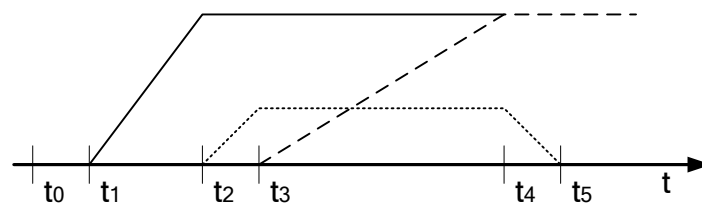


Abbildung 2: Zeitlicher Ablauf eines Roll-Back-Kommandos

Für die Bearbeitung eines Kommandos mit Transaktionsschutz werden, wie in Abbildung 2 gezeigt, die folgenden Zeitpunkte definiert:

- Zum Zeitpunkt t_0 werde das erste Bit der Kommando-APDU über die physikalische Schnittstelle (siehe Abbildung 1) 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 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 Abbildung 2 nicht dargestellt) habe die Smartcard das letzte Bit der Antwort-APDU über die physikalische Schnittstelle versendet.

(N034.500) K_COS

Dieses Dokument legt nicht fest, in welchem zeitlichen Zusammenhang t_e zu den anderen Zeiten steht. Damit KANN t_e zu einem beliebigen Zeitpunkt nach t_0 erfolgen.

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.

(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.

(N034.700) K_COS

Findet der Ausfall der Versorgungsspannung

- a. vor t_2 statt, so ist der Inhalt des Zwischenspeichers definitiv nicht auf gültig gesetzt, und damit ist ein Wechsel zum neuen Zustand unmöglich.
- b. 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.
- c. 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*). Falls ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, 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:

(N034.798) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass das aktuelle File zu aktivieren ist.

(N034.800) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 24 verwendet werden.

Tabelle 24: ACTIVATE aktuelles File

		Beschreibung
CLA	'00'	CLA Byte gemäß [ISO/IEC 7816-4]
INS	'44'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'00'	<i>mode</i> , 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:

(N034.810) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu aktivieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.

(N034.812) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.

(N034.814) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 25 verwendet werden.

Tabelle 25: 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'	<i>mode</i> , hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	<i>reference</i> 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:

(N034.820) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu aktivieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.

(N034.822) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt.

(N034.824) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 26 verwendet werden.

Tabelle 26: 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’	<i>mode</i> , hier: Schlüsselreferenz im Datenteil
P2	‘00’	-
Data	‘XX...XX’	‘83 – I2OS(OctetLength(<i>reference</i>), 1) – <i>reference</i> ’

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:

(N034.830) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu aktivieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.

(N034.832) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.

(N034.834) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 27 verwendet werden.

Tabelle 27: ACTIVATE Passwortobjekt

	Inhalt	Beschreibung
CLA	‘00’	CLA Byte gemäß [ISO/IEC 7816-4]
INS	‘44’	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	‘10’	<i>mode</i> , hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	‘XX’	<i>reference</i> auf ein Passwortobjekt

14.2.1.5 Antwort der Karte auf Aktivieren eines Files

Tabelle 28: ACTIVATE Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Aktivierung

Tabelle 29: 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 (69): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N034.900) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.1.6 Kommandoabarbeitung innerhalb der Karte

(N035.000) K_COS

- Das COS MUSS die ACTIVATE Varianten aus 14.2.1.1, 14.2.1.2, 14.2.1.3 und 14.2.1.4 unterstützen.
- Das COS KANN weitere ACTIVATE-Varianten
 - unterstützen oder
 - ablehnen.

(N035.100) K_COS

Falls 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.
 - andernfalls MUSS *affectedObject* gleich *currentFolder* gesetzt werden.
- '20' oder '21' besitzt, dann gilt *affectedObject* = SearchKey(
channelContext.currentFolder,
reference,
"Wildcard")
 - Gemäß 9.2.3 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.

- c. '10' besitzt, dann gilt *affectedObject* = *SearchPwd(currentFolder, reference)*. Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer *PasswordNotFound* terminieren.

(N035.110) K_COS

Falls *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer *VolatileKeyWithoutLCS* terminieren.

(N035.200) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.

(N035.300) K_COS

Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert

- a. „Termination state“ besitzt, dann MUSS das Kommando mit dem Trailer *Object-Terminated* terminieren.
- b. „Operational state (active)“ besitzt, dann MUSS als Trailer *NoError* verwendet werden.

(N035.400) K_COS

Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Operational state (active)“ gesetzt werden.

(N035.500) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer *UpdateRetryWarning* wählen.

(N035.600) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer *MemoryFailure* verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N035.700) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer *NoError* gewählt werden.

(N035.800) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 29 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 29 MUSS eine höhere Priorität als *UpdateRetryWarning* haben.
- c. *UpdateRetryWarning* MUSS eine höhere Priorität als *NoError* haben.

14.2.2 CREATE

(N035.900) K_COS

Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-9]

- a. unterstützen oder
- b. ablehnen.

Hinweis (70): 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*). Falls ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, 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:

(N035.998) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass das aktuelle File zu deaktivieren ist.

(N036.000) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 30 verwendet werden.

Tabelle 30: 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’	<i>mode</i> , 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:

(N036.010) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu deaktivieren ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.

(N036.012) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.

(N036.014) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 31 verwendet werden.

Tabelle 31: DEACTIVATE privates oder symmetrisches Schlüsselobjekt

	Inhalt	Beschreibung
--	--------	--------------

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'04'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'20'	<i>mode</i> , hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	<i>reference</i> 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:

(N036.020) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu deaktivieren ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.

(N036.022) K_externeWelt {K_Karte}

Der Parameter *reference* MUSS eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt enthalten.

(N036.024) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 32 verwendet werden.

Tabelle 32: 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'	<i>mode</i> , hier: Schlüsselreferenz im Datenteil
P2	'00'	-
Data	'XX...XX'	'83 – I2OS(OctetLength(<i>reference</i>), 1) – <i>reference</i> '

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:

(N036.030) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu deaktivieren ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.

(N036.032) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.

(N036.034) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 33 verwendet werden.

Tabelle 33: DEACTIVATE Passwortobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'04'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'10'	<i>mode</i> , hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	'XX'	<i>reference</i> auf ein Passwortobjekt

14.2.3.5 Antwort der Karte auf Deaktivieren eines Files

Tabelle 34: DEACTIVATE Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Deaktivierung

Tabelle 35: 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 (71): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N036.100) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.3.6 Kommandoabarbeitung innerhalb der Karte

(N036.200) K_COS

- Das COS MUSS die DEACTIVATE-Varianten aus 14.2.3.1, 14.2.3.2, 14.2.3.3 und 14.2.3.4 unterstützen.
- Das COS KANN weitere DEACTIVATE-Varianten
 - unterstützen oder
 - ablehnen.

(N036.300) K_COS

Falls der Parameter *mode* in der Kommandonachricht den Wert

- '00' besitzt und *channelContext.currentEF* (siehe (N029.900)m)

1. auf eine Datei zeigt, dann MUSS *affectedObject* gleich *currentEF* gesetzt werden.
2. andernfalls MUSS *affectedObject* gleich *currentFolder* gesetzt werden.
- b. '20' oder '21' besitzt, dann gilt *affectedObject* = *SearchKey(channelContext.currentFolder, reference, "WildCard")*
 -) . Gemäß 9.2.3 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.
- c. '10' besitzt, dann gilt *affectedObject* = *SearchPwd(currentFolder, reference)* . Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer *PasswordNotFound* terminieren.

(N036.310) K_COS

Falls *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer *VolatileKeyWithoutLCS* terminieren.

(N036.400) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.

(N036.500) K_COS

Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert

- a. „Termination state“ besitzt, dann MUSS das Kommando mit dem Trailer *Object-Terminated* terminieren.
- b. „Operational state (deactivated)“ besitzt, dann MUSS als Trailer *NoError* verwendet werden.

(N036.600) K_COS

Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Operational state (deactivated)“ gesetzt werden.

(N036.700) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer *UpdateRetryWarning* wählen.

(N036.800) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer *MemoryFailure* verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N036.900) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer *NoError* gewählt werden.

(N037.000) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 35 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 35 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:

(N037.098) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass das aktuelle File zu löschen ist.

(N037.100) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 36 verwendet werden.

Tabelle 36: 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’	<i>mode</i> , 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:

(N037.110) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu löschen ist, wobei eine ein Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.

(N037.112) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.

(N037.114) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 37 verwendet werden.

Tabelle 37: 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'	<i>mode</i> , hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	<i>reference</i> 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:

(N037.120) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu löschen ist, wobei eine acht oder zwölf Oktett lange Schlüsselreferenz im Datenteil der Kommandonachricht enthalten ist.

(N037.122) K_externeWelt {K_Karte}

Der Parameter *reference* MUSS eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt enthalten.

(N037.124) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 38 verwendet werden.

Tabelle 38: 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'	<i>mode</i> , hier: Schlüsselreferenz im Datenteil
P2	'00'	-
Data	'XX...XX'	'83 – I2OS(OctetLength(<i>reference</i>), 1) – <i>reference</i> '

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:

(N037.130) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Passwortobjekt zu löschen ist, wobei eine ein Oktett lange Passwortreferenz im Kommandoheader enthalten ist.

(N037.132) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.

(N037.134) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 39 verwendet werden.

Tabelle 39: DELETE Passwortobjekt

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'E4'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'10'	<i>mode</i> , hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	'XX'	<i>reference</i> auf ein Passwortobjekt

14.2.4.5 Antwort der Karte auf Löschen eines Files

Tabelle 40: DELETE Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Löschoperation

Tabelle 41: 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 (72): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N037.200) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.4.6 Kommandoabarbeitung innerhalb der Karte

(N037.300) K_COS

- Das COS MUSS die DELETE-Varianten aus 14.2.4.1, 14.2.4.2, 14.2.4.3 und 14.2.4.4 unterstützen.
- Das COS KANN weitere DELETE-Varianten
 - unterstützen oder
 - ablehnen.

(N037.400) K_COS

Falls 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.

2. andernfalls MUSS *affectedObject* gleich *currentFolder* gesetzt werden.
 - b. '20' oder '21' besitzt, dann gilt *affectedObject* = SearchKey(
 channelContext.currentFolder,
 reference,
 "WildCard"
). Gemäß 9.2.3 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.
 - c. '10' besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference*). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
- (N037.500) K_COS
Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.
- (N037.600) K_COS
Die Löschoperation MUSS mit Transaktionsschutz durchgeführt werden.
- (N037.690) K_COS
Falls 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.
- (N037.700) K_COS
Wenn *affectedObject* vom Typ
- a. Ordner ist, dann MUSS
 1. clearSecurityStatusFolder(*affectedObject*) ausgeführt werden und
 2. aus *dfSpecificPasswordList* alle Einträge entfernt werden, die zu *affectedObject* gehören und
 3. in *keyReferenceList* alle Einträge auf den Wert „leer“ gesetzt werden, die auf Schlüsselobjekte verweisen, die *affectedObject* zugeordnet sind und
 4. *currentEF* auf den Wert undefined gesetzt werden und
 5. *currentFolder* auf den Vater von *affectedObject* gesetzt werden und
 6. aus *allPublicKeyList* MÜSSEN alle Einträge entfernt werden, die *affectedObject* zugeordnet sind und
 7. *affectedObject* inklusive Subtree gelöscht werden.
 - b. Datei ist, dann MUSS
 1. *currentEF* auf den Wert undefined gesetzt werden und
 2. *affectedObject* gelöscht werden.
 - c. Symmetrisches Authentisierungsobjekt oder symmetrisches Kartenverbindungsobjekt ist, dann MUSS
 1. clearSecurityStatusKey(*affectedObject*) ausgeführt werden und
 2. *affectedObject* gelöscht werden.
 - d. Passwortobjekt ist, dann MUSS

1. `clearPasswordStatus(affectedObject)` ausgeführt werden und
 2. `affectedObject` gelöscht werden.
- e. sonst MUSS `affectedObject` gelöscht werden.

(N037.800) K_COS

Falls `affectedObject` vom Typ

- a. Ordner oder Datei ist, dann MUSS vormals von diesem Objekt allozierter Speicher so freigegeben werden, dass er zum Anlegen anderer Objekte verwendbar ist.
- b. Symmetrisches Authentisierungsobjekt oder Passwortobjekt ist, dann
 1. SOLL vormals von diesem Objekt allozierter Speicher so freigegeben werden, dass er zum Anlegen anderer Objekte verwendbar ist.
 2. KANN das COS das betroffen Objekt löschen, ohne dass allozierter Speicher freigegeben wird.

(N037.850) K_COS

Wenn `affectedObject` ein Schlüssel ist, der in einem Element von `keyReferenceList` eingetragen ist, dann KANN der zum gelöschten Schlüssel gehörende Eintrag in `keyReferenceList`

- a. gelöscht werden oder
- b. bestehen bleiben.

(N037.900) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer `UpdateRetryWarning` wählen.

(N038.000) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer `MemoryFailure` verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N038.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer `NoError` gewählt werden.

(N038.200) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 41 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 41 MUSS eine höhere Priorität als `UpdateRetryWarning` haben.
- c. `UpdateRetryWarning` MUSS eine höhere Priorität als `NoError` haben.

(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“.

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 14.2.5.2 übertragen. In der hier beschriebenen Variante enthält die APDU des LOAD APPLICATION-Kommandos zwei Parameter.

(N038.400) K_externeWelt {K_Karte}

Das CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist. Dies wird durch den Wert CLA = '10' codiert.

(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.

(N038.600) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 42 verwendet werden.

Tabelle 42: 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'	<i>cmdData</i>

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.

(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.

(N038.800) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 43 verwendet werden.

Tabelle 43: 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’	<i>cmdData</i>

14.2.5.3 Antwort der Karte auf Anlegen neues Objekt

Tabelle 44: LOAD APPLICATION Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘63 Cx’	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
‘90 00’	NoError	Erfolgreicher Ladevorgang

Tabelle 45: 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	Identifizier 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 (73): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N038.900) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.5.4 Kommandoabarbeitung innerhalb der Karte

(N039.000) K_COS

- a. Das COS MUSS die LOAD APPLICATION-Varianten aus 14.2.5.1 und 14.2.5.2 unterstützen.
- b. Das COS KANN weitere LOAD APPLICATION-Varianten
 1. unterstützen oder
 2. ablehnen.

(N039.010) K_COS

Falls das Attribut *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) den Wert „Termination state“ besitzt, KANN das Kommando mit dem Trailer InstructionNotSupported terminieren.

(N039.100) K_COS

Als *affectedObject* MUSS *channelContext.currentFolder* verwendet werden.

(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.

(N039.300) K_COS

Das Kommando, und damit eine eventuell aktive Chaining-Kette, KANN akzeptiert oder abgelehnt werden, wenn das neu anzulegende File

- a. ein Ordner ist und ein Attribut *applicationIdentifier* besitzt und dieser *applicationIdentifier* bereits einem anderen Ordner innerhalb des Objektsystems zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DfNameExists terminieren), oder
- b. ein Ordner ist und ein Attribut *fileIdentifier* besitzt und dieser *fileIdentifier* bereits einem anderen File innerhalb von *affectedObject* zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren), oder
- c. eine Datei ist, deren Attribut *fileIdentifier* bereits einem anderen File innerhalb von *affectedObject* zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren), oder
- d. eine Datei ist, deren Attribut *shortFileIdentifier* bereits einer anderen Datei innerhalb von *affectedObject* zugeordnet ist (falls abgelehnt, genau dann MUSS das Kommando mit dem Trailer DuplicatedObject terminieren).

(N039.400) K_COS

Das Kommando MUSS akzeptiert werden, wenn keine der Bedingungen aus (N039.300) zutrifft und das neu anzulegende Objekt

- a. ein Ordner ist und ein Attribut *applicationIdentifier* besitzt und dieser *applicationIdentifier* keinem anderen Ordner innerhalb des Objektsystems zugeordnet ist,
- b. ein Ordner ist und ein Attribut *fileIdentifier* besitzt und dieser *fileIdentifier* keinem anderen File innerhalb von *affectedObject* zugeordnet ist, oder
- c. eine Datei ist, deren Attribut *fileIdentifier* keinem anderen File innerhalb von *affectedObject* zugeordnet ist, oder
- d. eine Datei ist, deren Attribut *shortFileIdentifier* keiner anderen Datei innerhalb von *affectedObject* zugeordnet ist.

(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.

(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.

(N039.700) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N039.800) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N039.900) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N040.000) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 45 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 45 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

(N040.100) K_COS

Wenn dies das einzige oder letzte Kommando einer Command-Chaining-Kette ist und das neu angelegte Objekt ist

- a. ein Ordner, dann MUSS im Erfolgsfall *channelContext.currentFolder* auf den neu angelegten Ordner gesetzt werden. Dabei sind die Regeln zum Setzen des Kanalcontextes für den neuen Ordner zu berücksichtigen (siehe (N048.200)b).
- b. eine Datei, dann MUSS im Erfolgsfall *currentEF* auf die neu angelegte Datei gesetzt werden.

(N040.200) K_COS

Wenn das Kommando nicht erfolgreich verlief, dann DÜRFEN *currentFolder* und *currentEF* NICHT verändert werden, auch wenn

- a. ein LOAD APPLICATION-Kommando mit einem Trailer aus Tabelle 45 terminierte, sondern MÜSSEN denselben Wert besitzen, wie vor der Ausführung dieses LOAD APPLICATION-Kommandos.
- b. eine Command-Chaining-Kette abgebrochen wird (siehe (N029.874)), sondern MÜSSEN denselben Wert besitzen, wie vor dem Start dieser Command-Chaining-Kette.

(N040.300) K_COS

Wenn dieses LOAD APPLICATION-Kommando mit einem Trailer aus Tabelle 45 terminierte, dann MUSS ein eventuell aktives Command Chaining abgebrochen werden.

(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

- a. das LOAD APPLICATION-Kommando mit einem Trailer aus Tabelle 45 terminierte, oder
- b. das LOAD APPLICATION-Kommando während der Kommandobearbeitung durch einen Reset abgebrochen wird, oder

c. eine Command-Chaining-Kette abgebrochen wird (siehe (N029.874)).

Hinweis (74): Es ist denkbar, dass Anforderung (N039.500) wie folgt getestet wird:

- a. Ausgangspunkt sei eine Smartcard, deren Objektsystem nur eine sehr geringe (minimale) Anzahl von Objekten enthält.
- b. Im ersten Schritt werde per LOAD APPLICATION-Kommando eine Datei (transparent oder strukturiert wird zufällig bestimmt) mit 200 Oktett Nutzdaten angelegt.
- c. Schritt b wird so lange wiederholt, bis das LOAD APPLICATION-Kommando mit dem Trailer OutOfMemory terminiert.
- d. Im zweiten Schritt werden zwei der zuvor angelegten Dateien zufällig ausgewählt und gelöscht (DELETE). 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.
- e. Schritt d 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:

(N040.500) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.

(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.

(N040.700) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.

(N040.800) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 46 verwendet werden.

Tabelle 46: 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'	<i>selectionMode</i> = Ordnerselektion mit <i>applicationIdentifier</i> (hier leer)
P2	'0C'	<i>fileOccurrence</i> + <i>responseType</i> = 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:

(N040.900) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.

(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.

(N041.100) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.

(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.

(N041.300) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 47 verwendet werden.

Tabelle 47: 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'	<i>selectionMode</i> = Ordnerselektion mit <i>applicationIdentifier</i> (hier leer)
P2	'04'	<i>fileOccurrence</i> + <i>responseType</i> = first occurrence, Antwortdaten mit FCP
Le	<i>length</i>	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:

(N041.400) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.

(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.

(N041.600) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.

(N041.700) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 48 verwendet werden.

Tabelle 48: 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'	<i>selectionMode</i> = Ordnerselektion mit <i>applicationIdentifier</i> (hier leer)
P2	'0E'	<i>fileOccurrence</i> + <i>responseType</i> = 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:

(N041.800) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.

(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.

(N042.000) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.

(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.

(N042.200) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 49 verwendet werden.

Tabelle 49: 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'	<i>selectionMode</i> = Ordnerselektion mit <i>applicationIdentifier</i> (hier leer)
P2	'06'	<i>fileOccurrence</i> + <i>responseType</i> = next occurrence, FCP-Antwortdaten
Le	<i>length</i>	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:

(N042.300) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.

(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.

(N042.500) 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.

(N042.600) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.

(N042.700) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 50 verwendet werden.

Tabelle 50: 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'	<i>selectionMode</i> = Ordnerselektion mit <i>applicationIdentifier</i>
P2	'0C'	<i>fileOccurrence</i> + <i>responseType</i> = first occurrence, keine Antwortdaten
Data	'XX...XX'	<i>aid</i> , 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:

(N042.800) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.

(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.

(N043.000) 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.

(N043.100) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.

(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.

(N043.300) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 51 verwendet werden.

Tabelle 51: 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'	<i>selectionMode</i> = Ordnerselektion mit applicationIdentifier
P2	'04'	<i>fileOccurrence</i> + <i>responseType</i> = first occurrence, Antwortdaten mit FCP
Data	'XX...XX'	<i>aid</i> , Oktettstring, Anzahl Oktette aus dem Intervall [1, 16]
Le	<i>length</i>	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:

(N043.400) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '04' gewählt werden.

(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.

(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.

(N043.700) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.

(N043.800) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 52 verwendet werden.

Tabelle 52: 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’	<i>selectionMode</i> = Ordnerselektion mit applicationIdentifier
P2	‘0E’	<i>fileOccurrence</i> + <i>responseType</i> = next occurrence, keine Antwortdaten
Data	‘XX...XX’	<i>aid</i> , 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:

(N043.900) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = ‘04’ gewählt werden.

(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.

(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.

(N044.200) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = ‘04’ gewählt werden.

(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.

(N044.400) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 53 verwendet werden.

Tabelle 53: 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'	<i>selectionMode</i> = Ordnerselektion mit applicationIdentifier
P2	'06'	<i>fileOccurrence</i> + <i>responseType</i> = next occurrence, Antwortdaten mit FCP
Data	'XX...XX'	<i>aid</i> , Oktettstring, Anzahl Oktette aus dem Intervall [1, 16]
Le	<i>length</i>	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:

(N044.500) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '01' gewählt werden.

(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.

(N044.700) K_externeWelt {K_Karte}

Der Parameter *fid* enthält einen Oktettstring gemäß (N006.700). Im *currentFolder* wird nach einem Ordner mit dazu passendem Attribut *fileIdentifier* gesucht.

(N044.800) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.

(N044.900) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 54 verwendet werden.

Tabelle 54: 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'	<i>selectionMode</i> = Selektion eines Ordners mit fileIdentifier
P2	'0C'	<i>fileOccurrence</i> + <i>responseType</i> = first occurrence, keine Antwortdaten
Data	'XXXX'	<i>fid</i>

14.2.6.10 Use Case Selektieren, DF oder ADF, Antwortdaten mit FCP

In dieser Variante enthält die APDU des SELECT-Kommandos fünf Parameter:

(N045.000) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '01' gewählt werden.

(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.

(N045.200) K_externeWelt {K_Karte}

Der Parameter *fid* enthält einen Oktettstring gemäß (N006.700). Im *currentFolder* wird nach einem Ordner mit dazu passendem Attribut *fileIdentifier* gesucht.

(N045.300) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.

(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.

(N045.500) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 55 verwendet werden.

Tabelle 55: 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'	<i>selectionMode</i> = Ordnerselektion mit fileIdentifier
P2	'04'	<i>fileOccurrence</i> + <i>responseType</i> = first occurrence, Antwortdaten mit FCP
Data	'XXXX'	<i>fid</i>
Le	<i>length</i>	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:

(N045.600) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '03' gewählt werden.

(N045.700) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.

(N045.800) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 56 verwendet werden.

Tabelle 56: 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'	<i>selectionMode</i> = Selektion des eine Ebene höheren Ordners
P2	'0C'	<i>responseType</i> = 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:

(N045.900) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '03' gewählt werden.

(N046.000) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.

(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.

(N046.200) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 57 verwendet werden.

Tabelle 57: 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'	<i>selectionMode</i> = Selektion des eine Ebene höheren Ordners
P2	'04'	<i>responseType</i> = Antwortdaten mit FCP
Le	<i>length</i>	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:

(N046.300) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '02' gewählt werden.

(N046.400) K_externeWelt {K_Karte}

Der Parameter *fileOccurrence* bestimmt, welches File aus einer Liste von passen-

den Files gefunden wird. Für diesen Use Case MUSS *fileOccurrence* = '0' gewählt werden.

(N046.500) 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.

(N046.600) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '0C' gewählt werden.

(N046.700) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 58 verwendet werden.

Tabelle 58: 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'	<i>selectionMode</i> = Selektion einer Datei mit <i>fileIdentifier</i>
P2	'0C'	<i>fileOccurrence</i> + <i>responseType</i> = first occurrence, keine Antwortdaten
Data	'XXXX'	<i>fid</i>

14.2.6.14 Use Case Selektieren einer Datei, Antwortdaten mit FCP

In dieser Variante enthält die APDU des SELECT-Kommandos fünf Parameter:

(N046.800) K_externeWelt {K_Karte}

Der Parameter *selectionMode* bestimmt die Art der Suche. Für diesen Use Case MUSS *selectionMode* = '02' gewählt werden.

(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.

(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.

(N047.100) K_externeWelt {K_Karte}

Der Parameter *responseType* bestimmt die Art der Antwortdaten. Für diesen Use Case MUSS *responseType* = '04' gewählt werden.

(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.

(N047.300) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 59 verwendet werden.

Tabelle 59: 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'	<i>selectionMode</i> = Selektion einer Datei mit <i>fileIdentifier</i>
P2	'04'	<i>fileOccurrence</i> + <i>responseType</i> = first occurrence, Antwortdaten mit FCP
Data	'XXXX'	<i>fid</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.2.6.15 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 60: 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 <i>fileIdentifier</i> Selektion einer Datei mit <i>fileIdentifier</i> Selektion des eine Ebene höher liegenden Ordners Selektion eines Ordners mit (möglicherweise leerem) <i>applicationIdentifier</i>
P2	'04' '06' '0C' '0E'	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 <i>fid</i> P1 = '02': zwei Byte <i>fid</i> P1 = '03': abwesend P1 = '04': abwesend, oder bis zu 16 Oktette <i>aid</i>
Le	<i>length</i>	P2 = '04': Anzahl der erwarteten Oktette in den Antwortdaten P2 = '06': Anzahl der erwarteten Oktette in den Antwortdaten P2 = '0C': abwesend P2 = '0E': abwesend

14.2.6.16 Antwort der Karte auf Selektieren eines Files

Tabelle 61: 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 62: 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 (75): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N047.400) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.6.17 Kommandoabarbeitung innerhalb der Karte

Zur Beschreibung der Kommandobearbeitung werden folgende Definitionen eingeführt:

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*.

(N047.500) K_COS

a. Das COS MUSS die SELECT-Varianten aus 14.2.6.1, 14.2.6.2, 14.2.6.5, 14.2.6.6, 14.2.6.9, 14.2.6.10, 14.2.6.11, 14.2.6.12, 14.2.6.13 und 14.2.6.14 unterstützen.

b. Das COS KANN weitere SELECT-Varianten

1. unterstützen oder

2. ablehnen.

(N047.590) K_COS

Falls das Attribut *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) den Wert „Termination state“ besitzt, genau dann MUSS das Kommando mit dem Trailer *InstructionNotSupported* terminieren.

(N047.600) K_COS

Zugriffsregeln für das SELECT Kommando:

- a. Das COS KANN die Auswertung von Zugriffsregeln für das SELECT Kommando unterstützen.
- b. Das COS KANN als Zugriffsbedingung für das SELECT Kommando stets ALWAYS verwenden.

(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.

(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.

(N047.900) K_COS

Wenn der Parameter P1 den Wert '03' besitzt, und *currentFolder*

- a. ist gleich *root*, dann MUSS das Kommando mit dem Trailer FileNotFound terminieren.
- b. ist nicht *root*, dann ist *newFile* der im Vergleich zu *currentFolder* eine Ebene höher liegende Ordner (siehe (N020.000), (N020.100)).

(N048.000) K_COS

Wenn der Parameter P1 den Wert '04' besitzt, dann enthält die Kommandonachricht einen (möglicherweise leeren) Parameter *aid*.

- a. Im gesamten Objektsystem wird nach Ordnern gesucht, die ein Attribut *applicationIdentifier* besitzen, welches zu *aid* passt. Ein Attribut *applicationIdentifier*
 - 1. MUSS als passend betrachtet werden, wenn *applicationIdentifier* identisch zu *aid* ist.
 - 2. MUSS als passend betrachtet werden, wenn *aid* leer ist.
 - 3. dessen Most Significant Bytes identisch sind zu *aid*
 - i. KANN als passend betrachtet werden.
 - ii. KANN als unpassend betrachtet werden.
- b. 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.
- c. Hat P2 einen Wert aus der Menge {'04', '0C'}, dann MUSS *newFile* auf das erste Listenelement gesetzt werden.
- d. 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
 - i. KANN *newFile* auf irgendein Listenelement gesetzt werden, oder
 - ii. die Suche KANN erfolglos sein.

(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 Kanal-kontext (siehe Kapitel 12) NICHT verändert werden.

(N048.200) K_COS

Wenn

- a. *newFile* eine Datei ist, dann MUSS *currentEF* auf *newFile* gesetzt werden.
- b. *newFile* ein Ordner ist, dann MUSS
 - 1. *currentFolder* gleich *newFile* gesetzt werden.
 - 2. *currentEF* auf den Wert „undefiniert“ gesetzt werden.
 - 3. in allen Ordnern, die sowohl zu *path(oldFolder)* als auch zu *path(newFile)* gehören, *seldentifier* unverändert bleiben.
 - 4. in allen anderen Ordnern *seldentifier* auf den Wert 1 gesetzt werden.
 - 5. die Funktion *clearSecurityStatusFolder(folder)* für alle Ordner ausgeführt werden, die nicht zu *path(newFile)* gehören.
 - 6. Option_Kryptobox: die Funktion *clearSessionkeys()* ausgeführt werden, falls *SessionkeyContext.folderSessionkeys* nicht zu *path(newFile)* gehört.
 - 7. aus *dfSpecificPasswordList* MÜSSEN mittels *clearPasswordStatus(...)* alle Einträge entfernt werden, die nicht zu *path(newFile)* gehören.
 - 8. jedes Element von *keyReferenceList*, welches ein Schlüsselobjekt referenziert, das zu einem Ordner außerhalb von *path(newFile)* gehört, auf den Wert „leer“ gesetzt werden.

(N048.300) K_COS

Für das Datenfeld *rspData* der Antwortnachricht gilt:

- a. 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: Falls *OctetLength(FCP)*
 - 1. kleiner Nr gemäß (N027.200): *rspData* = FCP.
 - 2. sonst *rspData* = *Extract_MSByte(FCP, Nr)*.
- b. Andernfalls fehlt das Datenfeld der Antwortnachricht.

(N048.400) K_COS

Wenn der logische Wert von *newFile.lifeCycleStatus* (siehe (N020.600)) den Wert

- a. „Operational state (deactivated)“ hat, genau dann MUSS als Trailer FileDeactivated gewählt werden.
- b. „Termination state“ hat, genau dann MUSS als Trailer FileTerminated gewählt werden.

(N048.500) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N048.600) K_COS

Für die Priorität der Trailer gilt:

- Die Priorität der Trailer in Tabelle 62 ist herstellerspezifisch.
- Jeder Trailer in Tabelle 62 MUSS eine höhere Priorität als FileDeactivated haben.
- FileDeactivated MUSS eine höhere Priorität als NoError haben.

Hinweis (76): Gemäß den Regeln dieses Dokumentes ist es zulässig, in deaktivierten oder terminierten Ordnern 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:

(N048.700) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 63 verwendet werden.

Tabelle 63: TERMINATE CARD USAGE

		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 64: TERMINATE CARD USAGE Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Terminierung

Tabelle 65: Terminate Card Usage Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis (77): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N048.738) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.7.3 Kommandoabarbeitung innerhalb der Karte

(N048.740) K_COS

- a. Das COS MUSS die TERMINATE CARD USAGE-Variante aus 14.2.7.1 unterstützen.
- b. Das COS KANN weitere TERMINATE CARD USAGE-Varianten
 1. unterstützen oder
 2. ablehnen.

(N048.742) K_COS

affectedObject MUSS gleich *root* gesetzt werden.

(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.

(N048.748) K_COS

Falls 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.

(N048.752) K_COS

Wenn der physikalische Wert von *lifeCycleStatus* des Objektsystems den Wert „Termination state“ besitzt, dann MUSS als Trailer *NoError* verwendet werden.

(N048.756) K_COS

Es werden folgende Aktionen ausgeführt:

- a. Der physikalische Wert von *lifeCycleStatus* des Objektsystems (siehe (N019.900)i) MUSS mittels Transaktionsschutz auf den Wert „Termination state“ gesetzt werden.
- b. Bis auf den Basiskanal MÜSSEN alle anderen logischen Kanäle geschlossen werden.
- c. Im Basiskanal MUSS *channelContext* gemäß (N030.100) gesetzt werden.

(N048.760) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer *UpdateRetryWarning* wählen.

(N048.762) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- entweder als Trailer MemoryFailure verwendet werden,
- oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N048.764) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N048.766) K_COS

Für die Priorität der Trailer gilt:

- Die Priorität der Trailer in Tabelle 65 ist herstellerspezifisch.
- Jeder Trailer in Tabelle 65 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- 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:

(N048.800) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 66 verwendet werden.

Tabelle 66: TERMINATE DF aktueller Ordner

		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 der Karte

Tabelle 67: TERMINATE DF Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘63 Cx’	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
‘90 00’	NoError	Erfolgreiche Terminierung

Tabelle 68: TERMINATE DF Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis (78): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N048.838) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.8.3 Kommandoabarbeitung innerhalb der Karte

(N048.840) K_COS

- a. Das COS MUSS die TERMINATE DF-Variante aus 14.2.8.1 unterstützen.
- b. Das COS KANN weitere TERMINATE DF-Varianten
 1. unterstützen oder
 2. ablehnen.

(N048.842) K_COS

affectedObject MUSS gleich *currentFolder* gesetzt werden.

(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.

(N048.852) K_COS

Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert „Termination state“ besitzt, dann MUSS als Trailer *NoError* verwendet werden.

(N048.856) K_COS

Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Termination state“ gesetzt werden.

(N048.860) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer *UpdateRetryWarning* wählen.

(N048.862) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer *MemoryFailure* verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N048.864) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer *NoError* gewählt werden.

(N048.866) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 68 ist herstellerspezifisch.

- b. Jeder Trailer in Tabelle 68 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*). Falls ein Schlüsselobjekt oder ein Passwortobjekt vom Kommando betroffen ist, wird dieses durch eine Schlüssel- oder Passwortreferenz bestimmt, die in der Kommandonachrichtis 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.

(N048.900) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass *currentEF* zu terminieren ist.

(N048.903) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 69 verwendet werden.

Tabelle 69: 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’	<i>mode</i> , der Wert ‘00’ zeigt an, dass <i>currentEF</i> 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:

(N048.910) K_externeWelt {K_Karte}

Der Parameter *mode* zeigt an, dass ein Schlüsselobjekt zu terminieren ist, wobei eine Oktett lange Schlüsselreferenz im Kommandoheader enthalten ist.

(N048.912) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.

(N048.914) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 70 verwendet werden.

Tabelle 70: 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'	<i>mode</i> , hier: Ein Oktett lange Schlüsselreferenz im Parameter P2
P2	'XX'	<i>reference</i> 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:

(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.

(N048.922) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine acht oder zwölf Oktett lange Schlüsselreferenz mit beliebigem Inhalt.

(N048.924) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 71 verwendet werden.

Tabelle 71: 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'	<i>mode</i> , hier: Schlüsselreferenz im Datenteil
P2	'00'	-
Data	'XX...XX'	'83 –l2OS(OctetLength(<i>reference</i>), 1) – <i>reference</i> '

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:

(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.

(N048.932) K_externeWelt {K_Karte}

Der Parameter *reference* enthält eine Passwortreferenz. Wert und Codierung MÜSSEN gemäß (N072.800) gewählt werden.

(N048.934) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „In-

terpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 72 verwendet werden.

Tabelle 72: TERMINATE Passwortobjekt

	Inhalt	Beschreibung
CLA	‘00’	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	‘E8’	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	‘10’	<i>mode</i> , hier: Ein Oktett lange Passwortreferenz im Parameter P2
P2	‘XX’	<i>reference</i> auf ein Passwortobjekt

14.2.9.5 Antwort der Karte auf terminieren von Datei, Schlüssel- oder Passwortobjekt

Tabelle 73: TERMINATE Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘63 Cx’	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
‘90 00’	NoError	Erfolgreiche Terminierung

Tabelle 74: 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 (79): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N048.948) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.2.9.6 Kommandoabarbeitung innerhalb der Karte

(N048.950) K_COS

- Das COS MUSS die TERMINATE-Varianten aus 14.2.9.1, 14.2.9.2, 14.2.9.3 und 14.2.9.4 unterstützen.
- Das COS KANN weitere TERMINATE-Varianten
 - unterstützen oder
 - ablehnen.

(N048.954) K_COS

Falls der Parameter *mode* in der Kommandonachricht den Wert

- ‘00’ besitzt und *channelContext.currentEF* (siehe (N029.900)m)

1. unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
- b. '20' oder '21' besitzt, dann gilt *affectedObject* = SearchKey(*channelContext.currentFolder*,
reference,
"WildCard")
). Gemäß 9.2.3 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.
- c. '10' besitzt, dann gilt *affectedObject* = SearchPwd(*currentFolder*, *reference*). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.
- (N048.955) K_COS
Falls *affectedObject* zu einem Eintrag in *volatileCache* gehört, dann MUSS das Kommando mit dem Trailer VolatileKeyWithoutLCS terminieren.
- (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.
- (N048.960) K_COS
Wenn der physikalische Wert von *lifeCycleStatus* von *affectedObject* den Wert „Termination state“ besitzt, dann MUSS als Trailer NoError verwendet werden.
- (N048.963) K_COS
Der physikalische Wert von *lifeCycleStatus* von *affectedObject* MUSS mittels Transaktionsschutz auf den Wert „Termination state“ gesetzt werden.
- (N048.966) K_COS
Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
- (N048.969) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
 - a. entweder als Trailer MemoryFailure verwendet werden,
 - b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
- (N048.972) K_COS
Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
- (N048.975) K_COS
Für die Priorität der Trailer gilt:
 - a. Die Priorität der Trailer in Tabelle 74 ist herstellerspezifisch.
 - b. Jeder Trailer in Tabelle 74 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) oder schreibend (UPDATE BINARY) zuzugreifen.

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 keine Dateien adressieren, die nicht *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:

(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)).

(N049.100) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 75 verwendet werden.

Tabelle 75: ERASE BINARY, logical EOF unverändert, ohne *shortFileIdentifier*

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4], <i>positionLogicalEndOfFile</i> konstant
INS	'0E'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	$(offset - P2) / 256$, MSByte von <i>offset</i>

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:

(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.

(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.

(N049.400) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 76 verwendet werden.

Tabelle 76: ERASE BINARY, logical EOF unverändert, mit *shortFileIdentifier*

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4], <i>positionLogicalEndOfFile</i> konstant
INS	'0E'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'XX'	128 + <i>shortFileIdentifier</i> , das heißt '80' + <i>shortFileIdentifier</i>
P2	'XX'	offset

14.3.1.3 Antwort der Karte auf Löschen in transparenten EF

Tabelle 77: ERASE BINARY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Löschvorgang

Tabelle 78: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6B 00'	OffsetTooBig	Parameter <i>offset</i> in Kommando-APDU ist zu groß

Hinweis (80): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N049.500) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.3.1.4 Kommandoabarbeitung innerhalb der Karte

(N049.600) K_COS

- a. Das COS MUSS die ERASE BINARY-Varianten aus 14.3.1.1 und 14.3.1.2 unterstützen.
- b. Das COS KANN weitere ERASE BINARY-Varianten
 1. unterstützen oder
 2. ablehnen.

(N049.700) K_COS

Falls die APDU des ERASE BINARY-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N049.800) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N049.900) K_COS

Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N050.000) K_COS

Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.

(N050.100) K_COS

Falls *affectedObject.flagTransactionMode* den Wert

- a. 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.

- b. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.

(N050.190) K_COS

Falls *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.

(N050.200) K_COS

Falls *offset* kleiner als *affectedObject.positionLogicalEndOfFile* ist dann

- a. MÜSSEN die durch *offset* gekennzeichnete Stelle und alle weiteren Oktette in *affectedObject.body* auf den Wert '00' gesetzt werden.
- b. 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.

(N050.300) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N050.400) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N050.500) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N050.600) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 78 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 78 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 (81): Es ist aus funktionaler Sicht irrelevant, ob die in (N050.200) 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.

14.3.2 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:

(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)).

(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.

(N051.100) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 79 verwendet werden.

Tabelle 79: 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 <i>offset</i>
P2	'XX'	<i>offset</i> mod 256, LSByte von <i>offset</i>
Le	<i>length</i>	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:

(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.

(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.

(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.

(N051.500) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 80 verwendet werden.

Tabelle 80: 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 + <i>shortFileIdentifier</i> , das heißt '80' + <i>shortFileIdentifier</i>
P2	'XX'	<i>offset</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.3.2.3 Antwort der Karte auf Lesen in transparenten EF

Tabelle 81: 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 <i>Le</i> angefordert
'90 00'	NoError	Erfolgreiche Leseoperation

Tabelle 82: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6B 00'	OffsetTooBig	Parameter <i>offset</i> in Kommando-APDU ist zu groß

Hinweis (82): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N051.600) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.3.2.4 Kommandoabarbeitung innerhalb der Karte

(N051.700) K_COS

- Das COS MUSS die READ BINARY-Varianten aus 14.3.2.1 und 14.3.2.2 unterstützen.
- Das COS KANN weitere READ BINARY-Varianten
 - unterstützen oder
 - ablehnen.

(N051.800) K_COS

Falls die APDU des READ BINARY-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N051.900) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N052.000) K_COS

Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N052.100) K_COS

Wenn *offset* größer oder gleich *affectedObject.positionLogicalEndOfFile* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.

(N052.200) K_COS

Wenn *affectedObject.flagChecksum* den Wert True hat und die Daten von *affectedObject.body* inkonsistent zur Checksumme sind, genau dann MUSS

- a. entweder als Trailer CorruptDataWarning gewählt werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(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.

(N052.400) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N052.500) K_COS

Für das Datenfeld der Antwortnachricht gilt:

- a. Aus dem Oktettstring von *affectedObject.body* MÜSSEN die durch *offset* gekennzeichnete Position und die nachfolgenden Oktette übernommen werden.
- b. Es DÜRFEN NICHT mehr Oktette übernommen werden, als durch *Ne* angegeben.

- c. Die Übernahme der Oktette MUSS so gestoppt werden, dass weder das Oktett an der Position *positionLogicalEndOfFile* noch nachfolgende Oktette übernommen werden.

(N052.600) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 82 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 82 MUSS eine höhere Priorität als CorruptDataWarning haben.
- c. CorruptDataWarning MUSS eine höhere Priorität als EndOfFileWarning haben.
- d. 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

(N052.900) K_COS

Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]

- a. unterstützen oder
- b. ablehnen.

14.3.4 SET LOGICAL EOF

Hinweis (83): 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 Löschoption 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.

(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)).

(N052.932) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „In-

terpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 83 verwendet werden.

Tabelle 83: 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'	(<i>offset</i> – P2) / 256, MSByte von <i>offset</i>
P2	'XX'	<i>offset</i> mod 256, LSByte von <i>offset</i>

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.

(N052.934) K_externeWelt {K_Karte}

Die APDU des SET LOGICAL EOF-Kommandos enthält zwei Parameter:

- Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
- Der Parameter *offset* bestimmt, ab welcher Position gelöscht wird. Der Wert von *offset* MUSS eine ganze Zahl im Intervall [0, 255] = ['00', 'FF'] sein.

(N052.936) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 84 verwendet werden.

Tabelle 84: 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 + <i>shortFileIdentifier</i> , das heißt '80' + <i>shortFileIdentifier</i>
P2	'XX'	<i>offset</i>

14.3.4.3 Antwort der Karte auf Setzen logical EOF in transparenten EF

Tabelle 85: SET LOGICAL EOF Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Löschvorgang

Tabelle 86: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6B 00'	OffsetTooBig	Parameter <i>offset</i> in Kommando-APDU ist zu groß

Hinweis (84): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N052.938) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.3.4.4 Kommandoabarbeitung innerhalb der Karte

(N052.940) K_COS

- a. Das COS MUSS die SET LOGICAL EOF-Varianten aus 14.3.4.1 und 14.3.4.2 unterstützen.
- b. Das COS KANN weitere SET LOGICAL EOF-Varianten
 1. unterstützen oder
 2. ablehnen.

(N052.942) K_COS

Falls die APDU des SET LOGICAL EOF-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N052.944) K_COS

Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N052.946) K_COS

Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N052.948) K_COS

Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.

(N052.950) K_COS

Wenn *affectedObject.flagTransactionMode* den Wert

- a. 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.
- b. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.

(N052.952) K_COS

Falls *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.

(N052.954) K_COS

Falls *offset*

- a. kleiner als *affectedObject.positionLogicalEndOfFile* ist dann
 1. 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.
 2. MUSS *positionLogicalEndOfFile* auf den Wert von *offset* geändert werden.
Hinweis: Aus Sicherheitsgründen erscheint es sinnvoll, die durch offset gekennzeichnete Stelle und alle weiteren Oktette in affectedObject.body auf den Wert '00' zu setzen. Ob so eine Aktion erfolgt, ist mit den normativen Mitteln aus diesem Dokument nicht testbar, vergleiche auch (N054.500)a und (N055.258).
- b. größer gleich *positionLogicalEndOfFile* ist, dann DARF *positionLogicalEndOfFile* NICHT geändert werden.

(N052.956) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N052.958) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N052.960) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N052.962) K_COS

Für die Priorität der Trailer gilt:

- Die Priorität der Trailer in Tabelle 86 ist herstellerspezifisch.
- Jeder Trailer in Tabelle 86 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

Hinweis (85): Es ist aus funktionaler Sicht irrelevant, ob die in (N052.954) 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:

(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)).

(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.

(N053.200) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 87 verwendet werden.

Tabelle 87: 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 <i>offset</i>
P2	'XX'	<i>offset</i> mod 256, LSByte von <i>offset</i>
Data	'XX...XX'	<i>newData</i>

14.3.5.2 Use Case Schreiben mit *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des UPDATE BINARY-Kommandos drei Parameter:

(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.

(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.

(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.

(N053.600) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 88 verwendet werden.

Tabelle 88: 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 + <i>shortFileIdentifier</i> , das heißt '80' + <i>shortFileIdentifier</i>
P2	'XX'	<i>offset</i>
Data	'XX...XX'	<i>newData</i>

14.3.5.3 Antwort der Karte auf Schreiben in transparenten EF

Tabelle 89: UPDATE BINARY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreicher Schreibvorgang

Tabelle 90: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6A 84'	DataTooBig	Parameter <i>newData</i> ragt über das Dateieinde hinaus

'6B 00'	OffsetTooBig	Parameter offset in Kommando-APDU ist zu groß
---------	--------------	---

Hinweis (86): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N053.700) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.3.5.4 Kommandoabarbeitung innerhalb der Karte

(N053.800) K_COS

- a. Das COS MUSS die UPDATE BINARY-Varianten aus 14.3.5.1 und 14.3.5.2 unterstützen.
- b. Das COS KANN weitere UPDATE BINARY-Varianten
 1. unterstützen
 2. ablehnen.

(N053.900) K_COS

Falls die APDU des UPDATE BINARY-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N054.000) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N054.100) K_COS

Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N054.200) K_COS

Wenn *offset* größer oder gleich *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer OffsetTooBig terminieren.

(N054.300) K_COS

Wenn $(offset + OctetLength(newData))$ größer als *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer DataTooBig terminieren.

(N054.400) K_COS

Wenn *affectedObject.flagTransactionMode* den Wert

- a. True hat, genau dann MÜSSEN *affectedObject.body* und gegebenenfalls *affectedObject.positionLogicalEndOfFile* mit Transaktionsschutz geändert werden. Der Transaktionsschutz MUSS auch die Anpassung der Checksumme zu *affectedObject.body* umfassen.
- b. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.

(N054.490) K_COS

Falls *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 den Daten, die sie schützt,

- a. DARF NICHT zum Kommandoabbruch führen.
- b. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.

(N054.500) K_COS

- a. Falls *affectedObject.positionLogicalEndOfFile* kleiner als *offset* ist, dann MUSS das Oktett an der Position *positionLogicalEndOfFile* und alle nachfolgenden Oktette bis zur Position (*offset* – 1) auf den Wert '00' gesetzt werden.
- b. Die durch *offset* gekennzeichnete Stelle in *affectedObject.body* und die folgenden Oktette MÜSSEN durch *newData* ersetzt werden.
- c. Falls *positionLogicalEndOfFile* kleiner als (*offset* + *OctetLength(newData)*) ist, dann MUSS *positionLogicalEndOfFile* = (*offset* + *OctetLength(newData)*) gesetzt werden.
- d. Falls *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.

(N054.600) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N054.700) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N054.800) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N054.900) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 90 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 90 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:

(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.

(N055.205) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 91 verwendet werden.

Tabelle 91: 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 <i>shortFileIdentifier</i>
P2	'00'	fester Wert
Data	'XX...XX'	<i>newData</i>

14.3.6.2 Use Case Anfügen mit *shortFileIdentifier* in transparenten EF

In dieser Variante enthält die APDU des WRITE BINARY-Kommandos zwei Parameter:

(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.

(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.

(N055.226) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „In-

terpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 92 verwendet werden.

Tabelle 92: 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 + <i>shortFileIdentifier</i> , das heißt ‘80’ + <i>shortFileIdentifier</i>
P2	‘00’	fester Wert
Data	‘XX...XX’	<i>newData</i>

14.3.6.3 Antwort der Karte auf Anfügen in transparenten EF

Tabelle 93: WRITE BINARY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘63 Cx’	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
‘90 00’	NoError	Erfolgreicher Schreibvorgang

Tabelle 94: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
‘6A 84’	DataTooBig	Parameter <i>newData</i> ragt über das Dateende hinaus

Hinweis (87): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N055.240) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.3.6.4 Kommandoabarbeitung innerhalb der Karte

(N055.244) K_COS

- a. Das COS MUSS die WRITE BINARY-Varianten aus 14.3.6.1 und 14.3.6.2 unterstützen.
- b. Das COS KANN weitere WRITE BINARY-Varianten
 1. unterstützen oder
 2. ablehnen.

(N055.246) K_COS

Falls die APDU des WRITE BINARY-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N055.248) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N055.250) K_COS

Wenn *affectedObject* nicht vom Typ transparent EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N055.252) K_COS

Wenn $(positionLogicalEndOfFile + OctetLength(newData))$ größer als *affectedObject.numberOfOctet* ist, genau dann MUSS das Kommando mit dem Trailer Data-TooBig terminieren.

(N055.254) K_COS

Wenn *affectedObject.flagTransactionMode* den Wert

- a. 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.
- b. False hat, dann MUSS das COS entscheiden, ob *affectedObject.body* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.

(N055.256) K_COS

Falls *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 den Daten, die sie schützt,

- a. DARF NICHT zum Kommandoabbruch führen.
- b. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.

(N055.258) K_COS

- a. Die durch *positionLogicalEndOfFile* gekennzeichnete Stelle in *affectedObject.body* und die folgenden Oktette MÜSSEN durch *newData* ersetzt werden.

- b. Das Attribut *positionLogicalEndOfFile* MUSS um *OctetLength(newData)* inkrementiert werden.
- c. Falls *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.

(N055.260) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer *UpdateRetryWarning* wählen.

(N055.262) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer *MemoryFailure* verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N055.264) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer *NoError* gewählt werden.

(N055.266) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 94 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 94 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 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 *record-List* 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:

(N055.300) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N055.400) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.

(N055.500) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 95 verwendet werden.

Tabelle 95: 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'	<i>recordNumber</i>
P2	'04'	<i>mode</i> , 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:

(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.

(N055.700) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N055.800) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.

(N055.900) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 96 verwendet werden.

Tabelle 96: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> < 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:

(N056.000) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N056.100) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.

(N056.200) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 97 verwendet werden.

Tabelle 97: 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'	<i>recordNumber</i>
P2	'05'	<i>mode</i> , 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:

(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.

(N056.400) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N056.500) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.

(N056.600) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 98 verwendet werden.

Tabelle 98: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> < 3) + '05' Codierung '05' bedeutet „nutze Listenelemente ab P1“

14.4.1.5 Antwort der Karte auf Aktivieren eines Rekords

Tabelle 99: ACTIVATE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Aktivierung

Tabelle 100: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement <i>recordNumber</i> existiert nicht

Hinweis (88): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N056.700) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.1.6 Kommandoabarbeitung innerhalb der Karte

(N056.800) K_COS

- a. Das COS MUSS die ACTIVATE RECORD-Varianten aus 14.4.1.1, 14.4.1.2, 14.4.1.3 und 14.4.1.4 unterstützen.
- b. Das COS KANN weitere ACTIVATE RECORD-Varianten
 - 1. unterstützen oder
 - 2. ablehnen.

(N056.900) K_COS

Falls die APDU des ACTIVATE RECORD-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 - 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 - 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 - 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 - 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N057.000) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N057.100) K_COS

Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N057.200) K_COS

Wenn *affectedObject.flagRecordLifeCycleStatus* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer NoRecordLifeCycleStatus terminieren.

(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.

(N057.400) K_COS

Wenn *affectedObject.flagTransactionMode* den Wert

- a. True hat, genau dann MUSS der *lifeCycleStatus* mit Transaktionsschutz geändert werden.
- b. False hat, dann MUSS das COS entscheiden, ob der *lifeCycleStatus* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.

(N057.500) K_COS

Wenn *mode* = '04' ist und der physikalische Wert von *lifeCycleStatus* des durch re-

cordNumber adressierten *record* in *affectedObject.recordList* bereits den Wert „Operational state (active)“ besitzt, dann MUSS als Trailer NoError verwendet werden.

(N057.600) K_COS

Der physikalische Wert von *lifeCycleStatus* der durch *recordNumber* und *mode* adressierten *record* in *affectedObject.recordList* MÜSSEN auf den Wert „Operational state (active)“ gesetzt werden. Dabei gilt: Wenn *mode* den Wert

- a. '04' besitzt, dann ist nur das durch *recordNumber* adressierte Listenelement betroffen.
- b. '05' besitzt, dann ist das durch *recordNumber* adressierte Listenelement und alle folgenden betroffen.

(N057.900) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N058.000) K_COS

Für den Trailer der Antwort-APDU gilt:

- a. Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
 1. entweder als Trailer MemoryFailure verwendet werden,
 2. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
- b. Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
- c. Für die Priorität der Trailer gilt:
 1. Die Priorität der Trailer in Tabelle 100 ist herstellerspezifisch.
 2. Jeder Trailer in Tabelle 100 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*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des APPEND RECORD-Kommandos enthält einen Parameter:

(N058.300) K_externeWelt {K_Karte}

Der Parameter *recordData* enthält die Daten des neuen Rekords. Der Parameter *re-*

cordData ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *recordData* MUSS aus dem in (N007.700) definierten Bereich gewählt werden.

(N058.400) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 101 verwendet werden.

Tabelle 101: 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'	<i>recordData</i>

14.4.2.2 Use Case Anlegen neuer Rekords, mit *shortFileIdentifier*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des APPEND RECORD-Kommandos enthält zwei Parameter:

(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.

(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) definierten Bereich gewählt werden.

(N058.700) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 102 verwendet werden.

Tabelle 102: 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 <i>shortFileIdentifier</i> , das heißt <i>shortFileIdentifier</i> << 3
Data	'XX...XX'	<i>recordData</i>

14.4.2.3 Antwort der Karte auf Anlegen eines neuen Rekords

Tabelle 103: APPEND RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Hinzufügen eines Rekords

Tabelle 104: APPEND RECORD Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'67 00'	WrongRecordLength	<i>recordData</i> 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6A 84'	FullRecordList	Rekordliste lässt keine weiteren Elemente zu
'6A 84'	OutOfMemory	Zu viele Oktette in <i>recordData</i>

Hinweis (89): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N058.800) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.2.4 Kommandoabarbeitung innerhalb der Karte

(N058.900) K_COS

- Das COS MUSS die APPEND RECORD-Varianten aus 14.4.2.1 und 14.4.2.2 unterstützen.
- Das COS KANN weitere APPEND RECORD-Varianten
 - unterstützen oder
 - ablehnen.

(N059.000) K_COS

Falls die APDU des APPEND RECORD-Kommandos

- einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 - erfolgreich verlief, dann MUSS
 - affectedObject* auf dieses EF gesetzt werden und
 - currentEF* auf dieses EF gesetzt werden.
 - nicht erfolgreich verlief, genau dann MUSS

- i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer *FileNotFound* terminieren.
 - b. keinen *shortFileIdentifier* enthält
 - 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer *NoCurrentEF* terminieren, ansonsten
 - 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.
- (N059.100) K_COS
Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.
- (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. Der Inhalt wurde nach (N059.650)a verschoben.
- (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. Der Inhalt wurde nach (N059.650)b verschoben.
- (N059.600) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N059.650)d verschoben.
- (N059.650) K_COS
Es werden folgende Schritte ausgeführt:
- a. Wenn *affectedObject* vom Typ
 - 1. linear fixes EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer *WrongRecordLength* terminieren.
 - 2. zyklisches EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer *WrongRecordLength* terminieren.
 - 3. linear variables EF ist und
 - i. die Anzahl Oktette in *recordData* größer als *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer *WrongRecordLength* terminieren.
 - ii. die Anzahl Oktette in den Oktettstrings aller *record* von *recordList* nach durchgeführter Listenerweiterung größer als *affectedObject.numberOctet* wäre, genau dann MUSS das Kommando mit dem Trailer *OutOfMemory* terminieren.
 - b. Wenn *affectedObject.flagTransactionMode* den Wert
 - 1. *True* hat, genau 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 ist.
2. False hat, dann MUSS das COS entscheiden, ob *affectedObject.recordList* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
 - c. Falls *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.
 - d. Wenn *affectedObject* vom Typ
 1. linear fixes EF oder linear variables EF ist, dann wird ein neuer Rekord an das Ende von *affectedObject.recordList* angehängt.
 2. zyklische EF ist, dann wird ein neuer Rekord am Anfang von *affectedObject.recordList* eingefügt. Falls dadurch die Anzahl der Listenelemente größer als *affectedObject.maximumNumberOfRecords* wird, genau dann MUSS das letzte Element in *affectedObject.recordList* gelöscht werden.
 - e. 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.
 - f. Falls der neu angelegte *record* in *affectedObject.recordList* durch eine Checksumme geschützt ist, dann MUSS dessen Checksumme auf einen Wert gesetzt werden, der konsistent zum Inhalt des neuen Rekords ist.
 - g. Als Oktettstring des neuen Rekords MUSS *recordData* verwendet werden.
 - h. Falls *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. Der Inhalt wurde nach (N059.650)g verschoben.
- (N059.800) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N059.650)h verschoben.
- (N059.900) K_COS
Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.
- (N060.000) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
 - a. entweder als Trailer MemoryFailure verwendet werden,
 - b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
- (N060.100) K_COS
Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
- (N060.200) K_COS
Für die Priorität der Trailer gilt:
 - a. Die Priorität der Trailer in Tabelle 104 ist herstellerspezifisch.

- b. Jeder Trailer in Tabelle 104 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- c. 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 Parameter:

(N060.500) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N060.600) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.

(N060.700) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 105 verwendet werden.

Tabelle 105: 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'	<i>recordNumber</i>
P2	'04'	<i>mode</i> , 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:

(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.

(N060.900) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N061.000) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '04' gewählt werden.

(N061.100) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 106 verwendet werden.

Tabelle 106: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> < 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:

(N061.200) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N061.300) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.

(N061.400) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 107 verwendet werden.

Tabelle 107: 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'	<i>recordNumber</i>
P2	'05'	<i>mode</i> , 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:

(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.

(N061.600) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das erste betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N061.700) K_externeWelt {K_Karte}

Der Parameter *mode* bestimmt die Art der Aktion. Für diesen Use Case MUSS *mode* = '05' gewählt werden.

(N061.800) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 108 verwendet werden.

Tabelle 108: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> << 3) + '05' Codierung '05' bedeutet „nutze Listenelemente ab P1“

14.4.3.5 Antwort der Karte auf Deaktivieren eines Rekords

Tabelle 109: DEACTIVATE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Deaktivieren

Tabelle 110: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement <i>recordNumber</i> existiert nicht

Hinweis (90): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N061.900) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.3.6 Kommandoabarbeitung innerhalb der Karte

(N062.000) K_COS

- a. Das COS MUSS die DEACTIVATE RECORD-Varianten aus 14.4.3.1, 14.4.3.2, 14.4.3.3 und 14.4.3.4 unterstützen.
- b. Das COS KANN weitere DEACTIVATE RECORD-Varianten
 1. unterstützen
 2. ablehnen.

(N062.100) K_COS

Falls die APDU des DEACTIVATE RECORD-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N062.200) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N062.300) K_COS

Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N062.400) K_COS

Wenn *affectedObject.flagRecordLifeCycleStatus* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer NoRecordLifeCycleStatus terminieren.

(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.

(N062.600) K_COS

Wenn *affectedObject.flagTransactionMode* den Wert

- a. True hat, genau dann MUSS der *lifeCycleStatus* mit Transaktionsschutz geändert werden.
- b. False hat, dann MUSS das COS entscheiden, ob der *lifeCycleStatus* mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.

(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.

(N062.800) K_COS

Der physikalische Wert von *lifeCycleStatus* der durch *recordNumber* adressierten *record* in *affectedObject.recordList* MUSS auf den Wert „Operational state (deactivated)“ gesetzt werden. Dabei gilt: Wenn *mode* den Wert

- a. '04' besitzt, dann ist nur das durch *recordNumber* adressierte Listenelement betroffen.
- b. '05' besitzt, dann ist das durch *recordNumber* adressierte Listenelement und alle folgenden betroffen.

(N062.900) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N063.000) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N063.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N063.200) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 110 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 110 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- c. 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 (91): 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'. Ein READ RECORD Kommando für Rekord zwei liefert dann: '039000')

14.4.4.1 Use Case Löschen eines Rekords ohne *shortFileIdentifier*

In dieser Variante wird ein Rekord aus der Liste *recordList* gelöscht.

(N063.420) K_externeWelt {K_Karte}

Die APDU des DELETE RECORD-Kommandos enthält einen Parameter:

- Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N063.422) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 111 verwendet werden.

Tabelle 111: 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'	<i>recordNumber</i>
P2	'04'	Codierung für „nutze Listenelement P1“

14.4.4.2 Use Case Löschen eines Rekords mit *shortFileIdentifier*

In dieser Variante wird ein Rekord aus der Liste *recordList* gelöscht.

(N063.424) K_externeWelt {K_Karte}

Die APDU des DELETE RECORD-Kommandos enthält zwei Parameter:

- Der Parameter *shortFileIdentifier* wählt während der Kommandoabarbeitung ein EF aus. Der Wert von *shortFileIdentifier* MUSS aus dem in (N007.000) definierten Bereich gewählt werden.
- Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N063.426) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 112 verwendet werden.

Tabelle 112: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> < 3) + '04' Codierung '04' bedeutet „nutze Listenelement P1“

14.4.4.3 Antwort der Karte auf Entfernen eines Rekords

Tabelle 113: DELETE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Entfernen eines Rekords

Tabelle 114: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement <i>recordNumber</i> existiert nicht

Hinweis (92): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N063.428) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.4.4 Kommandoabarbeitung innerhalb der Karte

(N063.430) K_COS

- a. Das COS MUSS die DELETE RECORD-Varianten aus 14.4.4.1 und 14.4.4.2 unterstützen.
- b. Das COS KANN weitere DELETE RECORD-Varianten
 1. unterstützen
 2. ablehnen.

(N063.432) K_COS

Falls die APDU des DELETE RECORD-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N063.434) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N063.436) K_COS

Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(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.

(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.

(N063.442) 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.

(N063.446) K_COS

Der Rekord wird wie folgt aus der Liste *recordList* entfernt:

- a. Der durch *recordNumber* adressierte *record* MUSS aus *affectedObject.recordList* entfernt werden und
- b. 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.

(N063.448) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N063.450) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N063.452) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N063.454) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 114 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 114 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- c. UpdateRetryWarning MUSS eine höhere Priorität als NoError haben.

14.4.5 ERASE RECORD

Das Kommando ERASE RECORD ersetzt den Oktettstring eines bereits vorhandenen Listenelementes in *recordList* eines strukturierten EF durch einen Oktettstring, der nur Oktette mit dem Wert '00' besitzt. Das betroffene strukturierte EF wird vor der Operation ausgewählt. Dies geschieht entweder vor dem Senden dieses ERASE 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. Der Rekord selbst verbleibt in der Liste *recordList*. Die APDU des ERASE RECORD Kommandos enthält einen Parameter:

(N063.500) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N063.600) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 115 verwendet werden.

Tabelle 115: 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'	<i>recordNumber</i>
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. Der Rekord selbst verbleibt in der Liste *recordList*. Die APDU des ERASE RECORD-Kommandos enthält zwei Parameter:

(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.

(N063.800) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(N063.900) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 116 verwendet werden.

Tabelle 116: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> << 3) + '04' Codierung '04' bedeutet „nutze Listenelement P1“

14.4.5.3 Antwort der Karte auf Löschen des Inhaltes eines Rekords

Tabelle 117: ERASE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiches Löschen eines Rekordinhaltes

Tabelle 118: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement <i>recordNumber</i> existiert nicht

Hinweis (93): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N064.000) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.5.4 Kommandoabarbeitung innerhalb der Karte

(N064.100) K_COS

- a. Das COS MUSS die ERASE RECORD-Varianten aus 14.4.5.1 und 14.4.5.2 unterstützen.
- b. Das COS KANN weitere ERASE RECORD-Varianten
 1. unterstützen
 2. ablehnen.

(N064.200) K_COS

Falls die APDU des ERASE RECORD-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N064.300) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N064.400) K_COS

Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(N064.500) K_COS

Wenn *recordNumber* größer als die Anzahl der Listenelemente in *affectedOb-*

ject.recordList ist, genau dann MUSS das Kommando mit dem Trailer RecordNot-Found terminieren.

(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.

(N064.700) K_COS

Wenn *affectedObject.flagTransactionMode* den Wert

- a. True hat, genau 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 ist.
- b. False hat, dann MUSS das COS entscheiden, ob der Rekordinhalt mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.

(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.

(N064.800) K_COS

Der Rekordinhalt wird wie folgt gelöscht

- a. Es MÜSSEN alle Oktette im Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* auf den Wert '00' gesetzt werden und
- b. 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 und
- c. 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.

(N064.900) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N065.000) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N065.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N065.200) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 118 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 118 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- c. 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:

(N065.500) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(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.

(N065.700) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 119 verwendet werden.

Tabelle 119: 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’	<i>recordNumber</i>
P2	‘04’	Codierung für „nutze Listenelement P1“
Le	<i>length</i>	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:

(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.

(N065.900) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(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.

(N066.100) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 120 verwendet werden.

Tabelle 120: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> << 3) + '04' Codierung '04' bedeutet „nutze Listenelement P1“
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.4.6.3 Antwort der Karte auf Lesen in strukturierten EF

Tabelle 121: 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 122: 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 <i>shortFileIdentifier</i> adressiertes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement <i>recordNumber</i> existiert nicht

Hinweis (94): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N066.200) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.6.4 Kommandoabarbeitung innerhalb der Karte

(N066.300) K_COS

- a. Das COS MUSS die READ RECORD-Varianten aus 14.4.6.1 und 14.4.6.2 unterstützen.
- b. Das COS KANN weitere READ RECORD-Varianten
 1. unterstützen oder
 2. ablehnen.

(N066.400) K_COS

Falls die APDU des READ RECORD-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N066.500) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N066.600) K_COS

Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(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.

(N066.800) K_COS

Wenn der physikalische Wert von *lifeCycleStatus* des adressierten *record* den Zu-

stand „Operational state (deactivated)“ hat, genau dann MUSS das Kommando mit dem Trailer RecordDeactivated terminieren.

(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.

(N067.000) K_COS

Wenn das LeFeld der Kommando-APDU keine WildCard enthält und *length* größer als die Länge des adressierten Listenelementes *affectedObject.numberOfOctet* ist, genau dann MUSS als Trailer EndOfRecordWarning gewählt werden.

(N067.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N067.200) K_COS

Für das Datenfeld der Antwortnachricht gilt:

- a. Aus dem Oktettstring des adressierten *record* MÜSSEN vom ersten Oktett an die nachfolgenden Oktette übernommen werden.
- b. Es DÜRFEN NICHT mehr Oktette übernommen werden, als durch *length* angegeben.
- c. Die Übernahme MUSS am Ende von *record* stoppen.

(N067.300) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 122 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 122 MUSS eine höhere Priorität als CorruptDataWarning haben.
- c. CorruptDataWarning MUSS eine höhere Priorität als EndOfRecordWarning haben.
- d. 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:

(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) gewählt werden.

(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 kleiner oder gleich 255 Oktett sein.

(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.

(N067.900) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 123 verwendet werden.

Tabelle 123: 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'	<i>recordNumber</i>
P2	'04'	Codierung für „suche in Listenelement P1 und allen folgenden“
Data	'XX...XX'	<i>searchString</i>
Le	<i>length</i>	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:

(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.

(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) gewählt werden.

(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 kleiner oder gleich 255 Oktett sein.

(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.

(N068.400) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 124 verwendet werden.

Tabelle 124: 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'	<i>recordNumber</i>
P2	'XX'	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> << 3) + '04' '04' bedeutet: „suche in Listenelement P1 und allen folgenden“
Data	'XX...XX'	<i>searchString</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.4.7.3 Antwort der Karte auf Suchen in strukturierten EF

Tabelle 125: SEARCH RECORD Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>rspData</i>	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 126: 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 <i>shortFileIdentifier</i> ausgewähltes EF nicht gefunden
'6A 83'	RecordNotFound	Listenelement <i>recordNumber</i> existiert nicht

Hinweis (95): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N068.500) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.7.4 Kommandoabarbeitung innerhalb der Karte

(N068.600) K_COS

- a. Das COS MUSS die SEARCH RECORD-Varianten aus 14.4.7.1 und 14.4.7.2 unterstützen.
- b. Das COS KANN weitere SEARCH RECORD-Varianten
 1. unterstützen oder
 2. ablehnen.

(N068.700) K_COS

Falls die APDU des SEARCH RECORD-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N068.800) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N068.900) K_COS

Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(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.

(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.

(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.

(N069.300) K_COS

Wenn die Suche in einem Listenelement erfolgreich war, dann MUSS die Nummer des Rekords (siehe (N007.600)) in einem Oktett (gemäß I2OS(*recordNumber*, 1)) codiert zum Datenfeld *rspData* der Antwortnachricht hinzugefügt werden.

(N069.400) K_COS

Die Oktette im Datenfeld *rspData* MÜSSEN aufsteigend sortiert sein.

(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.

(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.

(N069.700) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N069.710) K_COS

Für das Datenfeld *rspData* der Antwortnachricht gilt: Falls OctetLength(*rspData*)

- a. kleiner Nr gemäß (N027.200): *rspData* = *rspData*.
- b. sonst *rspData* = Extract_MSByte(*rspData*, Nr).

(N069.800) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 126 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 126 MUSS eine höhere Priorität als CorruptDataWarning haben.
- c. CorruptDataWarning MUSS eine höhere Priorität als UnsuccessfulSearch haben.
- d. 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*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des UPDATE RECORD-Kommandos enthält zwei Parameter:

(N070.100) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(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) definierten Bereich gewählt werden.

(N070.300) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 127 verwendet werden.

Tabelle 127: 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’	<i>recordNumber</i>
P2	‘04’	Codierung für „nutze Listenelement P1“
Data	‘XX...XX’	<i>newData</i>

14.4.8.2 Use Case Rekordinhalt schreiben, mit *shortFileIdentifier*

Diese Variante stellt entweder das Ende einer Kommandokette dar, oder das einzige Kommando einer „Kette“. Die APDU des UPDATE RECORD-Kommandos enthält drei Parameter:

(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.

(N070.500) K_externeWelt {K_Karte}

Der Parameter *recordNumber* bestimmt das betroffene Listenelement. Der Wert von *recordNumber* MUSS konform zu (N007.600) gewählt werden.

(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) definierten Bereich gewählt werden.

(N070.700) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 128 verwendet werden.

Tabelle 128: 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’	<i>recordNumber</i>
P2	‘XX’	8 <i>shortFileIdentifier</i> + <i>mode</i> , das heißt (<i>shortFileIdentifier</i> < 3) + ‘04’ Codierung ‘04’ bedeutet „nutze Listenelement P1“
Data	‘XX...XX’	<i>newData</i>

14.4.8.3 Antwort der Karte auf Schreiben in strukturierten EF

Tabelle 129: UPDATE RECORD Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘63 Cx’	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
‘90 00’	NoError	Erfolgreicher Schreibvorgang

Tabelle 130: 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	<i>newData</i> 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 <i>shortFileIdentifier</i> ausgewähltes EF nicht gefunden
‘6A 83’	RecordNotFound	Listenelement <i>recordNumber</i> existiert nicht
‘6A 84’	OutOfMemory	Zu viele Oktette in <i>newData</i>

Hinweis (96): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N070.800) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.4.8.4 Kommandoabarbeitung innerhalb der Karte

(N070.900) K_COS

- a. Das COS MUSS die UPDATE RECORD-Varianten aus 14.4.8.1 und 14.4.8.2 unterstützen.
- b. Das COS KANN weitere UPDATE RECORD-Varianten
 - 1. unterstützen oder
 - 2. ablehnen.

(N071.000) K_COS

Falls die APDU des UPDATE RECORD-Kommandos

- a. einen *shortFileIdentifier* enthält, dann wird innerhalb von *currentFolder.children* nach einem EF mit diesem *shortFileIdentifier* gesucht. Falls die Suche
 - 1. erfolgreich verlief, dann MUSS
 - i. *affectedObject* auf dieses EF gesetzt werden und
 - ii. *currentEF* auf dieses EF gesetzt werden.
 - 2. nicht erfolgreich verlief, genau dann MUSS
 - i. *currentEF* unverändert bleiben und
 - ii. das Kommando mit dem Trailer FileNotFound terminieren.
- b. keinen *shortFileIdentifier* enthält
 - 1. und *currentEF* (siehe (N029.900)m) unbestimmt ist, genau dann MUSS das Kommando mit dem Trailer NoCurrentEF terminieren, ansonsten
 - 2. MUSS *affectedObject* gleich *currentEF* gesetzt werden.

(N071.100) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N071.200) K_COS

Wenn *affectedObject* nicht vom Typ strukturiertes EF ist, genau dann MUSS das Kommando mit dem Trailer WrongFileType terminieren.

(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.

(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. Der Inhalt wurde nach (N071.750)a verschoben.

(N071.600) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N071.750)b verschoben.

(N071.700) Diese Anforderung ist absichtlich leer. Der Inhalt wurde nach (N071.750)d verschoben.

(N071.750) K_COS

Es werden folgende Schritte ausgeführt:

- a. Wenn *affectedObject* vom Typ
 - 1. linear fixes EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer *WrongRecordLength* terminieren.
 - 2. zyklisches EF ist und die Anzahl Oktette in *recordData* ungleich *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer *WrongRecordLength* terminieren.
 - 3. linear variables EF ist und
 - i. die Anzahl Oktette in *recordData* größer als *affectedObject.maximumRecordLength* ist, genau dann MUSS das Kommando mit dem Trailer *WrongRecordLength* terminieren.
 - ii. die Anzahl Oktette in den Oktettstrings aller *record* von *affectedObject.recordList* nach durchgeführter Ersetzung größer als *affectedObject.numberOfOctet* wäre, genau dann MUSS das Kommando mit dem Trailer *OutOfMemory* terminieren.
- b. Wenn *affectedObject.flagTransactionMode* den Wert
 - 1. True hat, genau dann MUSS der Rekordeinhalt 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 ist.
 - 2. False hat, dann MUSS das COS entscheiden, ob der Rekordeinhalt mit oder ohne Transaktionsschutz (siehe 14.1) geändert wird.
- c. 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,
 - 1. DARF NICHT zum Kommandoabbruch führen.
 - 2. MUSS die Kommandobearbeitung gemäß (N031.940) stoppen.
- d. K_COS
 - 1. Der Oktettstring des durch *recordNumber* adressierten *record* in *affectedObject.recordList* wird durch *recordData* ersetzt.
 - 2. 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.
 - 3. 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.

(N071.800) K_COS

Falls *affectedObject* vom Typ linear variables EF ist, dann MÜSSEN alle folgenden Fälle unterstützt werden: *recordData* enthält im Vergleich zum *record*, der durch *recordNumber* adressiert wird

- a. weniger Oktette,

- b. gleich viele Oktette, oder
- c. mehr Oktette.

(N071.900) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N072.000) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N072.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N072.200) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 130 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 130 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

(N072.500) K_COS

Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]

- a. unterstützen oder
- b. ablehnen.

14.5 Zugriff auf Datenobjekte

Hinweis (97): Gemäß 8.7 sind die Kommandos dieses Kapitels nicht verpflichtend.

14.5.1 GET DATA

(N072.600) K_COS

Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]

- a. unterstützen oder
- b. ablehnen.

14.5.2 PUT DATA

(N072.700) K_COS

Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]

- a. unterstützen oder
- b. ablehnen.

14.6 Benutzerverifikation

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:

(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:

- a. Der Wert *location* = '00' MUSS verwendet werden, wenn ein globales Passwortobjekt betroffen ist (siehe (N020.800)).
- b. Der Wert *location* = '80' MUSS verwendet werden, wenn ein DF-spezifisches Passwortobjekt betroffen ist (siehe (N020.900)).
- c. Der Parameter *identifier* bestimmt das betroffene Passwortobjekt. Der Wert von *identifier* MUSS konform zu (N015.000) gewählt werden.
- d. 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:

(N072.900) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N073.000) K_externeWelt {K_Karte}

Der Parameter *oldSecret* enthält das alte Benutzergeheimnis.

(N073.100) K_externeWelt {K_Karte}

Der Parameter *newSecret* enthält das neue Benutzergeheimnis.

(N073.200) K_externeWelt {K_Karte}

Die Parameter *oldSecret* und *newSecret* MÜSSEN gemäß (N008.100) codiert sein.

(N073.300) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 131 verwendet werden.

Tabelle 131: 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'	<i>passwordReference</i>
Data	'XX...XX'	<i>oldSecret</i> <i>newSecret</i>

14.6.1.2 Use Case Setzen eines Benutzergeheimnisses

In dieser Variante enthält die APDU des CHANGE REFERENCE DATA-Kommandos zwei Parameter:

(N073.400) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N073.500) K_externeWelt {K_Karte}

Der Parameter *newSecret* enthält das neue Benutzergeheimnis.

(N073.600) K_externeWelt {K_Karte}

Der Parameter *newSecret* MUSS gemäß (N008.100) codiert sein.

(N073.700) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 132 verwendet werden.

Tabelle 132: 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'	<i>passwordReference</i>
Data	'XX...XX'	<i>newSecret</i>

14.6.1.3 Antwort der Karte auf Ändern eines Benutzergeheimnisses

Tabelle 133: CHANGE REFERENCE DATA Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	<i>oldSecret</i> ist falsch
'90 00'	NoError	Erfolgreiches Ändern des Benutzergeheimnisses

Tabelle 134: 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 Fehlbedienungszyklus
'69 85'	LongPassword	<i>newData</i> enthält ein zu langes Passwort
'69 85'	ShortPassword	<i>newData</i> enthält ein zu kurzes Passwort
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis (98): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N073.800) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.6.1.4 Kommandoabarbeitung innerhalb der Karte

(N073.900) K_COS

Das COS MUSS die CHANGE REFERENCE DATA-Varianten aus 14.6.1.1 und 14.6.1.2 unterstützen.

(N074.000) Diese Anforderung ist absichtlich leer.

(N074.100) K_COS

Das COS KANN weitere CHANGE REFERENCE DATA-Varianten

- a. unterstützen oder
- b. ablehnen.

(N074.200) K_COS

Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.

(N074.250) Diese Anforderung ist absichtlich leer.

(N074.300) K_COS

Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N074.400) K_COS

Wenn *affectedObject.retryCounter* den Wert null hat, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.

(N074.500) K_COS

Wenn die in *newSecret* codierte Ziffernfolge für das Attribut *secret* des Passwortobjektes eine Länge hat, die

- a. kleiner als *affectedObject.minimumLength* ist, genau dann MUSS das Kommando mit dem Trailer ShortPassword terminieren.
- b. größer als *affectedObject.maximumLength* ist, genau dann MUSS das Kommando mit dem Trailer LongPassword terminieren.

(N074.600) Diese Anforderung ist absichtlich leer.

Hinweis (99): 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.

(N074.700) K_COS

Wenn das Datenfeld der Kommandonachricht *oldSecret* enthält, genau dann MUSS das Attribut *affectedObject.secret* mit *oldSecret* verglichen werden.

- a. 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 Low-nibble des Trailers MUSS dabei auf den Wert 'F' gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*.
- b. Wenn der Vergleich erfolgreich ist, genau dann MUSS das Attribut *affectedObject.retryCounter* auf den Wert *affectedObject.startRetryCounter* gesetzt werden.
- c. Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.

(N074.710) K_COS

Das Attribut *affectedObject.secret* MUSS auf den in *newSecret* codierten Wert gesetzt werden.

(N074.720) K_COS

Das Attribut *affectedObject.transportStatus* MUSS auf den Wert *regularPassword* geändert werden (siehe (N009.500)).

(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. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.

(N075.000) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N075.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N075.200) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 134 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 134 MUSS eine höhere Priorität als WrongSecretWarning haben.
- c. WrongSecretWarning MUSS eine höhere Priorität als NoError haben.

(N075.300) K_COS

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. Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.
- b. *retryCounter* KANN in einer eigenen Transaktion zeitlich vor anderen Attributen geändert werden.
- c. *secret* KANN in einer eigenen Transaktion zeitlich vor *transportStatus* geändert werden.
- d. *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, falls das Benutzergeheimnis nicht korrekt ist. Die APDU des DISABLE VERIFICATION REQUIREMENT-Kommandos enthält zwei Parameter:

(N075.380) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N075.382) K_externeWelt {K_Karte}

Der Parameter *verificationData* enthält das Benutzergeheimnis.

(N075.384) K_externeWelt {K_Karte}

Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.

(N075.386) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 135 verwendet werden.

Tabelle 135: 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
P2	'XX'	<i>passwordReference</i>
Data	'XX...XX'	<i>verificationData</i>

14.6.2.2 Use Case Abschalten der Benutzerverifikation ohne Benutzergeheimnis

In dieser Variante enthält die APDU des DISABLE VERIFICATION REQUIREMENT-Kommandos einen Parameter:

(N075.400) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N075.500) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 136 verwendet werden.

Tabelle 136: 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
P2	'XX'	<i>passwordReference</i>

14.6.2.3 Antwort der Karte auf Abschalten der Benutzerverifikation

Tabelle 137: DISABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	<i>verificationData</i> ist falsch
'90 00'	NoError	Erfolgreiches Abschalten des Passwortobjektes

Tabelle 138: 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 Fehlbedienungszyklus
'69 85'	PasswordNotUsable	Passwort mit Transportschutz versehen

'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden
---------	------------------	--

Hinweis (100): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N075.600) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.6.2.4 Kommandoabarbeitung innerhalb der Karte

(N075.700) K_COS

- Das COS MUSS die DISABLE VERIFICATION REQUIREMENT-Variante aus 14.6.2.1 und 14.6.2.2 unterstützen.
- Das COS KANN weitere DISABLE VERIFICATION REQUIREMENT-Varianten
 - unterstützen oder
 - ablehnen.

(N075.800) K_COS

Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.

(N075.900) K_COS

Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N076.000) Diese Anforderung ist absichtlich leer.

(N076.100) K_COS

Falls der Parameter P1 den Wert

- '00' besitzt, dann MÜSSEN folgende Schritte abgearbeitet werden:
 - Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.
 - Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.
 - Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.
 - Wenn der Vergleich fehlschlägt, genau dann MUSS
 - affectedObject.retryCounter* um eins dekrementiert werden und
 - clearPasswordStatus(*affectedObject*) ausgeführt werden und
 - 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*.
 - Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.
 - 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.
- b. '01' besitzt, dann MUSS das Attribut *affectedObject.flagEnabled* auf den Wert False gesetzt werden.
- (N076.110) K_COS
Alle persistenten Änderungen in (N076.100) MÜSSEN mit Transaktionsschutz ausgeführt werden.
- (N076.120) K_COS
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. Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.
 - b. *retryCounter* KANN in einer eigenen Transaktion zeitlich vor anderen Attributen geändert werden.
- (N076.200) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.
- (N076.300) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
- a. entweder als Trailer MemoryFailure verwendet werden,
 - b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
- (N076.400) Diese Anforderung ist absichtlich leer.
- (N076.500) K_COS
Für die Wahl des Trailers gilt: Falls *affectedObject.flagEnabled* den Wert
- a. False besitzt, genau dann MUSS als Trailer NoError verwendet werden
 - b. True besitzt, dann ist die Priorität der Trailer in Tabelle 138 herstellerspezifisch.
 - c. Jeder Trailer in Tabelle 138 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 nicht korrekt ist. Die APDU des ENABLE VERIFICATION REQUIREMENT-Kommandos enthält zwei Parameter:

(N076.580) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N076.582) K_externeWelt {K_Karte}

Der Parameter *verificationData* enthält das Benutzergeheimnis.

(N076.584) K_externeWelt {K_Karte}

Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.

(N076.586) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 139 verwendet werden.

Tabelle 139: 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'	<i>passwordReference</i>
Data	'XX...XX'	<i>verificationData</i>

14.6.3.2 Use Case Einschalten der Benutzerverifikation ohne Benutzergeheimnis

In dieser Variante enthält die APDU des ENABLE VERIFICATION REQUIREMENT-Kommandos einen Parameter:

(N076.600) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N076.700) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 140 verwendet werden.

Tabelle 140: 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'	<i>passwordReference</i>

14.6.3.3 Antwort der Karte auf Einschalten der Benutzerverifikation

Tabelle 141: ENABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	<i>verificationData</i> ist falsch
'90 00'	NoError	Erfolgreiches Einschalten des Passwortobjektes

Tabelle 142: 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 (101): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N076.800) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.6.3.4 Kommandoabarbeitung innerhalb der Karte

(N076.900) K_COS

- Das COS MUSS die ENABLE VERIFICATION REQUIREMENT-Variante aus 14.6.3.1 und 14.6.3.2 unterstützen.
- Das COS KANN weitere ENABLE VERIFICATION REQUIREMENT-Varianten
 - unterstützen oder
 - ablehnen.

(N077.000) K_COS

Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.

(N077.100) K_COS

Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N077.200) Diese Anforderung ist absichtlich leer.

(N077.300) K_COS

Falls der Parameter P1 den Wert

- '00' besitzt, dann MÜSSEN folgende Schritte abgearbeitet werden:
 - Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.
 - Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.
 - Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.
 - Wenn der Vergleich fehlschlägt, genau dann MUSS
 - affectedObject.retryCounter* um eins dekrementiert werden und
 - 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*.
 - ii. Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.
 - iii. 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.
 - b. '01' besitzt, dann MUSS das Attribut *affectedObject.flagEnabled* auf den Wert True gesetzt werden.
- (N077.310) K_COS
Alle persistenten Änderungen in (N077.300) MÜSSEN mit Transaktionsschutz ausgeführt werden.
- (N077.320) K_COS
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. Alle Attribute SOLLEN gemeinsam in einer Transaktion geändert werden.
 - b. *retryCounter* KANN in einer eigenen Transaktion zeitlich vor anderen Attributen geändert werden.
- (N077.400) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.
- (N077.500) K_COS
Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS
 - a. entweder als Trailer MemoryFailure verwendet werden,
 - b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.
- (N077.600) Diese Anforderung ist absichtlich leer.
- (N077.700) K_COS
Für die Wahl des Trailers gilt: Falls *affectedObject.flagEnabled* den Wert
 - a. True besitzt, genau dann MUSS als Trailer NoError verwendet werden.
 - b. False besitzt, dann ist die Priorität der Trailer in Tabelle 142 herstellerspezifisch.
 - c. Jeder Trailer in Tabelle 142 MUSS eine höhere Priorität als WrongSecretWarning haben.

14.6.4 GET PIN STATUS

Hinweis (102): 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.

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:

(N077.800) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N077.900) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 143 verwendet werden.

Tabelle 143: GET PIN STATUS

	Inhalt	Beschreibung
CLA	'80'	CLA-Byte gemäß [ISO/IEC 7816-4] wird hier „proprietary“ angezeigt
INS	'20'	Instruction Byte (dies ist derselbe Wert wie beim Kommando VERIFY)
P1	'00'	–
P2	'XX'	<i>passwordReference</i>

14.6.4.2 Antwort der Karte auf Auslesen des PIN Status

Tabelle 144: 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	Passwortobjekt eingeschaltet, Passwortobjekt ohne Transportschutz (das bedeutet regularPassword), Wert des Fehlbedienungs Zählers enthalten im Least Significant Nibble
'90 00'	NoError	Passwortobjekt eingeschaltet, Sicherheitszustand gesetzt

Tabelle 145: GET PIN STATUS Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis (103): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N078.000) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.6.4.3 Kommandoabarbeitung innerhalb der Karte

(N078.100) K_COS

- a. Das COS MUSS die GET PIN STATUS-Variante aus 14.6.4.1 unterstützen.
- b. Das COS KANN weitere GET PIN STATUS-Varianten
 1. unterstützen oder
 2. ablehnen.

(N078.200) K_COS

Es gilt *affectedObject* = *SearchPwd(currentFolder, passwordReference)*. Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer *PasswordNotFound* terminieren.

(N078.300) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.

(N078.400) K_COS

Falls das Attribut *affectedObject.flagEnabled* (siehe (N015.700)) den Wert *False* besitzt, dann MUSS als Trailer *PasswordDisabled* verwendet werden.

(N078.500) K_COS

Falls *affectedObject.flagEnabled* (siehe (N015.700)) den Wert *True* besitzt und *affectedObject* in *globalPasswordList* (siehe (N029.900)i) oder in *dfSpecificPassword-List* (siehe (N029.900)j) enthalten ist und dort das Attribut *securityStatusEvaluation-Counter* (siehe (N029.900)k) einen Wert ungleich null besitzt, dann MUSS als Trailer *NoError* verwendet werden.

(N078.600) K_COS

Falls 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 Tabelle 144 zu ersetzen.

(N078.700) K_COS

Falls 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*.

(N078.800) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 145 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 145 MUSS eine höhere Priorität als *PasswordDisabled* haben.

- c. PasswordDisabled MUSS eine höhere Priorität als NoError haben.
- d. NoError MUSS eine höhere Priorität als TransportStatus haben.
- e. TransportStatus MUSS eine höhere Priorität als RetryCounter haben.

Hinweis (104): Gemäß (N078.600), (N078.700) und (N078.800) ist es nicht möglich den Fehlbedienungs-zähler auszulesen, wenn Transportschutz besteht.

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 und ohne PUK
- mit und ohne neuen Wert für das Attribut *secret* des Passwortobjektes.

Die Variante „mit PUK“ wird typischerweise verwendet, wenn eine „nicht technische“ Instanz (etwa der Kartenbesitzer) das Recht zur Durchführung dieser Aktion besitzt.

Die Variante „ohne PUK“ wird typischerweise verwendet, wenn eine technische Instanz (etwa das CMS) diese Aktion durchzuführen hat. 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 Fehlbedienungs-zählers 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 Entsperrten mit PUK, mit neuem Geheimnis

In dieser Variante enthält die APDU des RESET RETRY COUNTER-Kommandos drei Parameter:

(N078.900) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N079.000) K_externeWelt {K_Karte}

Der Parameter *PUK* enthält ein Geheimnis, welches diese Aktion autorisiert.

(N079.100) K_externeWelt {K_Karte}

Der Parameter *newSecret* enthält das neue Benutzergeheimnis.

(N079.200) K_externeWelt {K_Karte}

Die Parameter *PUK* und *newSecret* MÜSSEN gemäß (N008.100) codiert sein.

(N079.300) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 146 verwendet werden.

Tabelle 146: 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 <i>PUK</i> und neues Benutzergeheimnis
P2	'XX'	<i>passwordReference</i>
Data	'XX...XX'	<i>PUK</i> <i>newSecret</i>

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:

(N079.400) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N079.500) K_externeWelt {K_Karte}

Der Parameter *PUK* enthält ein Geheimnis, welches diese Aktion autorisiert.

(N079.600) K_externeWelt {K_Karte}

Der Parameter *PUK* MUSS gemäß (N008.100) codiert sein.

(N079.700) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 147 verwendet werden.

Tabelle 147: 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 <i>PUK</i>
P2	'XX'	<i>passwordReference</i>
Data	'XX...XX'	<i>PUK</i>

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:

(N079.800) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N079.900) K_externeWelt {K_Karte}

Der Parameter *newSecret* enthält das neue Benutzergeheimnis.

(N080.000) K_externeWelt {K_Karte}

Der Parameter *newSecret* MUSS gemäß (N008.100) codiert sein.

(N080.100) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 148 verwendet werden.

Tabelle 148: 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’	<i>passwordReference</i>
Data	‘XX...XX’	<i>newSecret</i>

14.6.5.4 Use Case Entsperrten ohne PUK, ohne neues Geheimnis

In dieser Variante enthält die APDU des RESET RETRY COUNTER-Kommandos einen Parameter:

(N080.200) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N080.300) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 149 verwendet werden.

Tabelle 149: 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’	<i>passwordReference</i>

14.6.5.5 Antwort der Karte auf Entsperrten eines Benutzergeheimnisses

Tabelle 150: RESET RETRY COUNTER Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘63 Cx’	WrongSecretWarning	PUK ist falsch
‘90 00’	NoError	Erfolgreiches Rücksetzen des Fehlbedienungszählers

Tabelle 151: 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 (105): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N080.400) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.6.5.6 Kommandoabarbeitung innerhalb der Karte

(N080.500) K_COS

- Das COS MUSS die RESET RETRY COUNTER-Varianten aus 14.6.5.1, 14.6.5.2, 14.6.5.3 und 14.6.5.4 unterstützen.
- Das COS KANN weitere RESET RETRY COUNTER-Varianten
 - unterstützen oder
 - ablehnen.

(N080.600) K_COS

Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.

(N080.700) K_COS

Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N080.800) 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.
- größer als *affectedObject.maximumLength* ist, genau dann MUSS das Kommando mit dem Trailer LongPassword terminieren.

(N080.900) K_COS

Mittels clearPasswordStatus(*affectedObject*) MUSS der Sicherheitsstatus zurückgesetzt werden.

(N081.000) K_COS

Wenn das Kommandodatenfeld einen Parameter *PUK* enthält, genau dann MUSS dieser mit dem Attribut *affectedObject.PUK* verglichen werden.

- a. Wenn *affectedObject.pukUsage* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer CommandBlocked terminieren.
- b. *affectedObject.pukUsage* MUSS mit Transaktionsschutz um eins dekrementiert werden.
- c. 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 (106): 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.

(N081.100) K_COS

Das Attribut *affectedObject.retryCounter* MUSS auf den Wert *affectedObject.startRetryCounter* gesetzt werden.

(N081.200) K_COS

Wenn das Kommandodatenfeld einen Parameter *newSecret* enthält, genau dann MUSS das Attribut

- a. *affectedObject.secret* auf den in *newSecret* codierten Wert gesetzt werden.
- b. *affectedObject.transportStatus* auf den Wert „regularPassword“ (siehe (N009.500)) gesetzt werden, falls dieses Attribut einen anderen Wert besitzt.

(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. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.

(N081.500) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N081.600) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N081.700) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 151 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 151 MUSS eine höhere Priorität als WrongSecretWarning haben.
- c. WrongSecretWarning MUSS eine höhere Priorität als NoError haben.

(N081.800) K_COS

Wenn die Ausführung dieses Kommandos durch einen Reset abgebrochen wird, dann gilt für die möglichen Änderungen von *secret*, *transportStatus* und *retryCounter*:

- a. Alle Änderungen SOLLEN gemeinsam in einer Transaktion geändert werden.
- b. Jede Änderung KANN in einer eigenen Transaktion durchgeführt werden. In diesem Fall 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:

(N081.900) K_externeWelt {K_Karte}

Der Parameter *passwordReference* referenziert das von der Aktion betroffene Passwort und MUSS gemäß (N072.800) gewählt werden.

(N082.000) K_externeWelt {K_Karte}

Der Parameter *verificationData* enthält das Benutzergeheimnis.

(N082.100) K_externeWelt {K_Karte}

Der Parameter *verificationData* MUSS gemäß (N008.100) codiert sein.

(N082.200) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 152 verwendet werden.

Tabelle 152: 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'	<i>passwordReference</i>
Data	'XX...XX'	<i>verificationData</i>

14.6.6.2 Antwort der Karte auf Vergleich eines Benutzergeheimnisses

Tabelle 153: VERIFY Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'63 Cx'	WrongSecretWarning	<i>verificationData</i> ist falsch
'90 00'	NoError	Erfolgreicher Vergleich

Tabelle 154: VERIFY Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'65 81'	MemoryFailure	Schreibvorgang nicht erfolgreich
'69 82'	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt
'69 83'	PasswordBlocked	Abgelaufener Fehlbedienungszyklus

Trailer	Inhalt	Beschreibung
'69 85'	PasswordNotUsable	Passwort mit Transportschutz versehen
'6A 88'	PasswordNotFound	Adressiertes Passwort wurde nicht gefunden

Hinweis (107): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N082.300) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.6.6.3 Kommandoabarbeitung innerhalb der Karte

(N082.400) K_COS

- a. Das COS MUSS die VERIFY-Variante aus 14.6.6.1 unterstützen.
- b. Das COS KANN weitere VERIFY-Varianten
 1. unterstützen oder
 2. ablehnen.

(N082.500) K_COS

Es gilt *affectedObject* = SearchPwd(*currentFolder*, *passwordReference*). Falls die Passwortsuche mit einem Fehler abbricht, genau dann MUSS das Kommando mit dem Trailer PasswordNotFound terminieren.

(N082.600) K_COS

Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N082.700) K_COS

Wenn *affectedObject.retryCounter* den Wert null besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordBlocked terminieren.

(N082.800) K_COS

Wenn *affectedObject.transportStatus* nicht den Wert *regularPassword* besitzt, genau dann MUSS das Kommando mit dem Trailer PasswordNotUsable terminieren.

(N082.900) K_COS

Das Attribut *affectedObject.secret* MUSS mit *verificationData* verglichen werden.

- a. 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 Low-nibble des Trailers MUSS dabei auf den Wert 'F' gesetzt werden, wenn *affectedObject.retryCounter* größer als fünfzehn ist, andernfalls auf den Wert von *affectedObject.retryCounter*.
- b. Wenn die Vergleichsoperation durch einen Reset abgebrochen wird, dann MUSS *affectedObject.retryCounter* um eins dekrementiert werden.
- c. Wenn der Vergleich erfolgreich ist, genau dann MUSS

1. setPasswordStatus(*affectedObject*) ausgeführt werden und
2. das Attribut *affectedObject.retryCounter* auf den Wert *affectedObject.startRetryCounter* gesetzt werden und
3. die Änderung von *affectedObject.retryCounter* mit Transaktionsschutz ausgeführt werden.

(N083.000) Diese Anforderung ist absichtlich leer. In einer früheren Version war der Trailer UpdateRetryWarning zulässig. Dies ist nun nicht mehr der Fall.

(N083.100) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer MemoryFailure verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N083.200) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N083.300) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 154 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 154 MUSS eine höhere Priorität als WrongSecretWarning haben.
- c. 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 14.9.9.4, 14.9.9.5, 14.9.9.6). Die Antwort der externen Instanz auf das von der Karte generierte Token ist als Parameter in der Kommandonachricht enthalten.

Hinweis (108): 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 (109): 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 eGK, HBA und SMC aus. Dies erscheint vertretbar, da es derzeit für die Verwendung des Protokolls T=0 keine Anforderung gibt.

14.7.1.1 Use Case externe Authentisierung ohne Antwortdaten

In dieser Variante enthält die APDU des EXTERNAL AUTHENTICATE-Kommandos einen Parameter:

(N083.400) 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* ist abhängig von der mittels (N101.300) oder (N101.800) ausgewählten *algld*. Wenn *algld* gleich

- elcRoleCheck ist, dann MUSS unter Hinzuziehung der Domainparameter des öffentlichen Schlüssels gelten: $8 \text{ OctetLength}(\text{cmdData}) = 2 \text{ domainParameter}.\tau$.
- Option_RSA_CVC, rsaRoleCheck ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.
- Option_DES, rsaSessionkey4SM ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.

(N083.402) K_externeWelt {K_Karte}, Option_Kryptobox

Der Parameter *cmdData* enthält die Antwort der externen Instanz. Der Parameter *cmdData* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *cmdData* ist abhängig von der mittels (N101.300) oder (N101.800) ausgewählten *algld*. Wenn *algld* gleich

- aesSessionkey4TC ist, dann MUSS *cmdData* gleich 120 Oktett lang sein.
- Option_DES, desSessionkey4TC ist, dann MUSS *cmdData* gleich 104 Oktett lang sein.
- Option_DES, rsaSessionkey4TC ist, dann MUSS *cmdData* genauso viele Oktette enthalten, wie der Modulus des Authentisierungsschlüssels.

(N083.500) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 155 verwendet werden.

Tabelle 155: 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'	<i>cmdData</i>

14.7.1.2 Use Case externe Authentisierung mit Antwortdaten

In dieser Variante enthält die APDU des MUTUAL AUTHENTICATE-Kommandos zwei Parameter:

(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* ist in abhängig von der mittels (N102.300) ausgewählten *algld*. Wenn *algld* gleich

- aesSessionkey4SM ist, dann MUSS *cmdData* 120 Oktett lang sein.
- Option_DES, desSessionkey4SM ist, dann MUSS *cmdData* 104 Oktett lang sein.

(N083.700) K_externeWelt {K_Karte}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N083.800) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 156 verwendet werden.

Tabelle 156: 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'	<i>cmdData</i>
Le	'00'	length, Anzahl der erwarteten Oktette in den Antwortdaten

14.7.1.3 Antwort der Karte auf externe Authentisierung

Tabelle 157: EXTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>authData</i>	Authentisierungsdaten (konditional, abhängig von <i>algId</i>)
Trailer	Inhalt	Beschreibung
'63 00'	AuthenticationFailure	Authentisierung fehlgeschlagen
'90 00'	NoError	Erfolgreiche Authentisierung

Tabelle 158: 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 angegebenen Algorithmus nicht
'6A 88'	KeyNotFound	Authentisierungsschlüssel nicht gefunden
'6A 88'	PrKNotFound	Entschlüsselungsschlüssel nicht gefunden

Hinweis (110): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N083.900) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.7.1.4 Kommandoabarbeitung innerhalb der Karte

(N084.000) Kommandovarianten

a. K_COS

Das COS MUSS die EXTERNAL AUTHENTICATE-Variante aus 14.7.1.1 und 14.7.1.2 unterstützen.

b. Das COS KANN weitere EXTERNAL AUTHENTICATE-Varianten

1. unterstützen oder
2. ablehnen.

(N084.100) K_COS

Wenn *RND.ICC* den Wert "NoRandom" besitzt (siehe (N099.300), (N029.900)b und Kapitel 15), genau dann MUSS das Kommando mit dem Trailer NoRandom terminieren.

(N084.200) K_COS

Wenn *channelContext.keyReferenceList.externalAuthenticate*

a. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.

b. nicht leer ist, dann wird *affectedObject* = SearchKey(
channelContext.currentFolder,
keyReferenceList.externalAuthenticate.keyReference,
keyReferenceList.externalAuthenticate.algID

) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.

(N084.220) K_COS

Das Kommando MUSS mit dem Trailer WrongRandomLength terminieren, falls *RND.ICC* eine Länge von

a. acht Oktetten besitzt und *keyReferenceList.externalAuthenticate.algID* Element der folgenden Menge ist: {aesSessionkey4*, elcRoleCheck}.

b. 16 Oktetten besitzt und *keyReferenceList.externalAuthenticate.algID* Element der folgenden Menge ist: {desSessionkey4* (Option_DES), rsa* (Option_RSA_CVC)}.

(N084.280) K_COS

Das COS KANN den Sicherheitszustand mittels clearSecurityStatus-Key(*affectedObject*) vor der Auswertung der Zugriffsregel (siehe (N084.300)) oder vor der Auswertung der Kommandodaten (siehe (N084.400)) zurücksetzen. In diesem Fall ist das Löschen des Sicherheitszustandes gemäß (N084.500) nicht erforderlich.

(N084.300) K_COS

Falls AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.

(N084.400) K_COS

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 wird *cmdData* wie folgt verarbeitet: Wenn *channelContext.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert

a. *elcRoleCheck* besitzt, dann gilt:

1. Falls *affectedObject.expirationDate* kleiner als *pointInTime* ist (siehe (N019.900j)), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.

2. *cmdData* MUSS wie folgt auf *R* und *S* aufgeteilt werden:

i. *cmdData* = *R* || *S*.

ii. *OctetLength(R)* = *OctetLength(S)*.

3. Setze *PuK* = *affectedObject.publicKey*.

4. Setze *hash* = I2OS(
OS2I(*RND.ICC* || *iccsn8* || *elcRoleCheck*),
PuK.domainParameter.τ / 8
).

Hinweis: iccsn8 ist in (N019.900)c definiert.

5. Setze *out* = ELC_VER_SIG(*PuK*, *R*, *S*, *hash*).

Falls *out* den Wert *False* besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.

b. Option_RSA_CVC, *rsaRoleCheck* besitzt, dann gilt mit *PuK* = *affectedObject.publicKey*

1. Setze *M2* = *RND.ICC* || *iccsn8*. *Hinweis: iccsn8 ist in (N019.900)c definiert.*

2. (*out*, *M*) = RSA_ISO9796_2_DS1_VERIFY(*PuK*, *cmdData*, *M2*)

Falls diese Operation mit einem Fehler abbricht, oder *out* gleich *False* ist, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.

c. Option_DES, *rsaSessionkey4SM* besitzt, dann gilt *authData* = "" (leerer Oktettstring) und es werden folgende Schritte ausgeführt:

1. Wenn *channelContext.keyReferenceList.internalAuthenticate*

i. leer ist, genau dann MUSS das Kommando mit dem Trailer NoPrkReference terminieren.

ii. nicht leer ist, dann wird *tmpObject* = SearchKey(
channelContext.currentFolder,
keyReferenceList.internalAuthenticate.keyReference,
keyReferenceList.externalAuthenticate.algID
) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler

A. *keyNotFound* meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.

B. *notSupported* meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.

2. Das COS KANN eine Zugriffsregelprüfung für den privaten Schlüssel in *tmpObject* gemäß *AccessRuleEvaluation(tmpObject, CLA, INS, P1, P2)* durchführen oder unterlassen (siehe Hinweis (111):).
3. Setze $tmpData = cmdData^{PrK.d} \bmod PrK.n$,
mit $PrK = tmpObject.privateKey$
4. Setze $M2 = RND.ICC \parallel iccsn8$
5. $(out, M) = RSA_ISO9796_2_DS1_VERIFY(PuK, tmpData, M2)$
Falls diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer *AuthenticationFailure* terminieren.
6. *M* wird aufgeteilt in $M = M1 \parallel M2$. Die 64 LSByte in *M1* MÜSSEN als *KD.e* an den Secure Messaging Layer übergeben werden (siehe 13.1).

(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 wird *cmdData* wie folgt weiterverarbeitet:

Wenn *channel-Context.keyReferenceList.externalAuthenticate.algorithmIdentifier* den Wert

- a. *aesSessionkey4TC* besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
 1. *cmdData* wird wie folgt aufgeteilt:
 - i. $cmdData = C1 \parallel MAC1$
 - ii. $OctetLength(MAC1) = 8$.
 2. $out = VerifyCMAC_IsoPadding(affectedObject.macKey, MAC1, C1)$
 3. Falls *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. $R = RND.ext \parallel ICCSN8.ext \parallel RND.int \parallel ICCSN8.int \parallel KD.e$
 - 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, falls
 - 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 13.1).

- b. Option_DES, desSessionkey4TC besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
1. *cmdData* wird wie folgt aufgeteilt:
 - i. $cmdData = C1 \parallel MAC1$
 - ii. $OctetLength(MAC1) = 8$.
 2. $out = VERIFY_Retail_MAC(affectedObject.macKey, MAC1, C1)$
 3. Falls *out* den Wert INVALID besitzt, dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren, sonst MÜSSEN die folgenden Schritte durchgeführt werden:
 4. $R = 3TDES_CBC_DEC(affectedObject.encKey, 0, C1)$
 5. *R* wird wie folgt aufgeteilt:
 - i. $R = RND.ext \parallel ICCSN8.ext \parallel RND.int \parallel ICCSN8.int \parallel KD.e$
 - 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, falls
 - 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 13.1).
- c. Option_DES, rsaSessionkey4TC besitzt, dann MUSS im Rahmen der Kommandobearbeitung identisch zu rsaSessionkey4SM verfahren werden.

(N084.410) K_COS

Die Antwort der externen Instanz ist in *cmdData* enthalten. Falls die Kommando APDU ein LeFeld enthält, mithin also ein Case 4 gemäß 11.7.4 vorliegt, dann handelt es sich um die MUTUAL AUTHENTICATE Variante des Kommandos. In diesem Fall wird *cmdData* wie folgt weiterverarbeitet, wobei *authData* berechnet wird:

Wenn *channelContext.keyReferenceList.externalAuthenticate.algID* den Wert

- a. aesSessionkey4SM besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
1. *cmdData* wird wie folgt aufgeteilt:
 2. $cmdData = C1 \parallel MAC1$
 3. $OctetLength(MAC1) = 8$.
 4. $out = VerifyCMAC_IsoPadding(affectedObject.macKey, MAC1, C1)$
 5. Falls *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. $S = \text{RND.ext} \parallel \text{ICCSN8.ext} \parallel \text{RND.int} \parallel \text{ICCSN8.int} \parallel \text{KD.e}$
 - iii. Es sei $16 = \text{OctetLength}(\text{RND.ext})$
 - iv. Es sei $8 = \text{OctetLength}(\text{ICCSN8.ext})$
 - v. Es sei $16 = \text{OctetLength}(\text{RND.int})$
 - vi. Es sei $8 = \text{OctetLength}(\text{ICCSN8.int})$
 - vii. Es sei $64 = \text{OctetLength}(\text{KD.e})$
7. Das Kommando MUSS mit AuthenticationFailure terminieren, falls
 - 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 $\text{KD.i} = \text{RAND}(64)$
9. Setze $R = \text{RND.int} \parallel \text{ICCSN8.int} \parallel \text{RND.ext} \parallel \text{ICCSN8.ext} \parallel \text{KD.i}$
10. Setze $C2 = \text{AES_CBC_ENC}(\text{affectedObject.encKey}, 0, R)$
11. Setze $\text{MAC2} = \text{CalculateCMAC_IsoPadding}(\text{affectedObject.macKey}, C2)$
12. Setze $\text{authData} = C2 \parallel \text{MAC2}$
13. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).
- b. Option_DES, desSessionkey4SM besitzt, dann MÜSSEN folgende Schritte in der angegebenen Reihenfolge ausgeführt werden:
 1. cmdData wird wie folgt aufgeteilt:
 2. $\text{cmdData} = C1 \parallel \text{MAC1}$
 3. $\text{OctetLength}(\text{MAC1}) = 8$.
 4. $\text{out} = \text{VERIFY_Retail_MAC}(\text{affectedObject.macKey}, \text{MAC1}, C1)$. Falls 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. $S = \text{3DES_CBC_DEC}(\text{affectedObject.encKey}, 0, C1)$
 - ii. S wird wie folgt aufgeteilt:
 - iii. $S = \text{RND.ext} \parallel \text{ICCSN8.ext} \parallel \text{RND.int} \parallel \text{ICCSN8.int} \parallel \text{KD.e}$
 - iv. Es sei $8 = \text{OctetLength}(\text{RND.ext})$
 - v. Es sei $8 = \text{OctetLength}(\text{ICCSN8.ext})$
 - vi. Es sei $8 = \text{OctetLength}(\text{RND.int})$
 - vii. Es sei $8 = \text{OctetLength}(\text{ICCSN8.int})$
 - viii. Es sei $64 = \text{OctetLength}(\text{KD.e})$
 6. Das Kommando MUSS mit AuthenticationFailure terminieren, falls
 - 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 $R = RND.int \parallel ICCSN8.int \parallel RND.ext \parallel ICCSN8.ext \parallel KD.i$

9. Setze $C2 = 3DES_CBC_ENC(affectedObject.encKey, 0, R)$

10. Setze $MAC2 = CALCULATE_Retail_MAC(affectedObject.macKey, C2)$

11. Setze $authData = C2 \parallel MAC2$

12. Die Oktettstrings KD.e und KD.i MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).

(N084.500) K_COS

Falls das Kommando mit dem Trailer AuthenticationFailure terminiert, genau dann MUSS `clearSecurityStatusKey(affectedObject)` ausgeführt werden.

(N084.600) K_COS

Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS `setSecurityStatus(affectedObject)` ausgeführt werden.

(N084.700) K_COS

Als Datenfeld der Antwortnachricht MUSS der (möglicherweise leere) Oktettstring *authData* verwendet werden.

(N084.800) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N084.900) K_COS

Für die Priorität der Trailer gilt:

a. Die Priorität der Trailer in Tabelle 158 ist herstellerspezifisch.

b. Jeder Trailer in Tabelle 158 MUSS eine höhere Priorität als NoError haben.

(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 (111): Im Rahmen dieser Spezifikation sind Authentisierungssequenzen nicht unterbrechbar (siehe (N104.600)). Das Authentisierungsprotokoll *rsaSessionkey4SM* gemäß 15.4.3 führt zunächst eine 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 (112): 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 des Authentisierungsprotokolls 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 Gegenseitige Authentisierung mittels PACE für Endnutzerkarten

Dieser Abschnitt behandelt die Kommandonachrichten für die in Abbildung 7 mit "COSa PICC" bezeichnete Seite, die in [BSI-TR-03110-2#3.2] "MRTD Chip (PICC)" genannt wird. Die Kommandonachrichten für die andere Seite werden in 14.7.2.4 beschrieben. Nach erfolgreichem Abschluss dieses Authentisierungsprotokolls liegen in der Karte Session-keys 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.

(N085.000) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.001) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 159 verwendet werden.

Tabelle 159: 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'	<i>length</i> , 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.

(N085.002) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- 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.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.003) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 160 verwendet werden.

Tabelle 160: 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'	<i>length</i> , 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.

(N085.004) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- 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.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.005) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 161 verwendet werden.

Tabelle 161: 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(~PK _{2PCD}))', DER codiertes Datenfeld
Le	'00'	<i>length</i> , 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.

(N085.006) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

- Der Parameter T_{PCD} ist ein Oktettstring der Länge acht mit beliebigem Inhalt.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.007) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 162 verwendet werden.

Tabelle 162: 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'	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten

14.7.2.2 Gegenseitige Authentisierung 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.

(N085.010) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei 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 *keyRef* MUSS zwölf Oktett lang sein. Er enthält eine Schlüsselreferenz für einen öffentlichen Schlüssel, der mittels CV-Zertifikat importiert wurde.
- c. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.012) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 163 verwendet werden.

Tabelle 163: 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 - <i>keyRef</i>)'
Le	'00'	<i>length</i> , 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.

(N085.014) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:

- a. Der Parameter *ephemeralPK_oponent* MUSS einen Punkt auf derselben elliptischen Kurve enthalten, wie der öffentliche Schlüssel, der in Schritt 1 selektiert wurde.

(N085.016) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 164 verwendet werden.

Tabelle 164: 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 - <i>ephemeralPK_oponent</i>)', DER codiertes Datenfeld

14.7.2.3 Authentisierung für asynchrone, symmetrische Kartenadministration

14.7.2.3.1 Use Case Authentisierung 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.

(N085.020) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.022) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 165 verwendet werden.

Tabelle 165: GENERAL AUTHENTICATE, asynchrone, sym. Admin., 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'	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten

14.7.2.3.2 Use Case Authentisierung 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.

(N085.024) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:

- Der Parameter *cmdData* MUSS 76 Oktett lang sein.

(N085.026) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 166 verwendet werden.

Tabelle 166: GENERAL AUTHENTICATE, asynchrone, sym. Admin., 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 Gegenseitige Authentisierung mittels PACE für Sicherheitsmodule

Dieser Abschnitt behandelt die Kommandonachrichten für die in Abbildung 7 mit "COSb PCD" bezeichnete Seite, die in [BSI-TR-03110-2#3.2] "Terminal (PCD)" genannt wird. Die Kommandonachrichten für die andere Seite werden in 14.7.2.1 beschrieben. Nach erfolgreichem Abschluss dieses Authentisierungsprotokolls liegen in der Karte 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.

(N085.030) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS zwei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.031) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 167 verwendet werden.

Tabelle 167: 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'	<i>length</i> , 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.

(N085.032) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- Der Parameter z ist ein Oktettstring der Länge 16 mit beliebigem Inhalt.
- Der Parameter CAN ist ein OktettString mit einer Länge aus dem Intervall [1, 16] und beliebigem Inhalt.

(N085.033) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 168 verwendet werden.

Tabelle 168: 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.

(N085.034) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- 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.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.035) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 169 verwendet werden.

Tabelle 169: 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(~PK1 _{PICC}))', DER codiertes Datenfeld
Le	'00'	<i>length</i> , 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.

(N085.036) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS drei Parameter enthalten:

- Der Parameter Chaining Bit im CLA-Byte zeigt an, dass diese Kommando-APDU nicht die letzte einer Command-Chaining-Kette ist.
- Der Parameter ~PK2_{PICC} ist ein Punkt, der so gewählt werden MUSS, dass bei der Decodierung in (N085.066)d.1 kein Fehler auftritt.
- Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardShort sein.

(N085.037) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 170 verwendet werden.

Tabelle 170: 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(~PK2 _{PICC}))', DER codiertes Datenfeld
Le	'00'	<i>length</i> , 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.

(N085.038) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:

- Der Parameter T_{PICC} ist ein Oktettstring der Länge acht mit beliebigem Inhalt.

(N085.039) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 168 verwendet werden.

Tabelle 171: 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 Authentisierung für asynchrone, asymmetrische Kartenadministration

14.7.2.5.1 Use Case Authentisierung 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.

14.7.2.5.2 Use Case Authentisierung 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.

(N085.040) K_externeWelt {K_Karte}

Die APDU des GENERAL AUTHENTICATE-Kommandos MUSS einen Parameter enthalten:

- 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.

(N085.041) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3S Kommando-APDU MÜSSEN die Angaben aus Tabelle 172 verwendet werden.

Tabelle 172: GENERAL AUTHENTICATE, asynchrone, asym. Admin., 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 - <i>cmdData</i>)' '7C – 820139 - (82 - 820135 - <i>cmdData</i>)' '7C – 82013B - (82 - 820177 - <i>cmdData</i>)'

falls brainpoolP256r1
falls brainpoolP384r1
falls brainpoolP512r1

14.7.2.6 Antwort der Karte auf generelle Authentisierung

Tabelle 173: GENERAL AUTHENTICATE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	responseData	Antwortdaten vorhanden
-	-	keine Antwortdaten
Trailer	Inhalt	Beschreibung
'63 00'	AuthenticationFailure	Authentisierung fehlgeschlagen
'90 00'	NoError	Erfolgreiche Authentisierung

Tabelle 174: 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 Szenarios nicht erfüllt
'6A 80'	NumberScenarioWrong	Scenario wurde bereits geladen
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegebenen Algorithmus nicht
'6A 88'	PrKNotFound	privaten Schlüssel nicht gefunden
'6A 88'	KeyNotFound	Authentisierungsschlüssel nicht gefunden

Hinweis (113): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N085.048) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.7.2.7 Kommandoabarbeitung innerhalb der Karte

(N085.050) Kommandounterstützung

- a. K_COS
Das COS MUSS die GENERAL AUTHENTICATE-Varianten aus 14.7.2.2, 14.7.2.3 und 14.7.2.5 unterstützen.
- b. K_COS, Option_kontaktlose_Schnittstelle
Das COS MUSS die GENERAL AUTHENTICATE-Variante aus 14.7.2.1 unterstützen.
- c. K_COS, Option_PACE_PCD
Das COS MUSS die GENERAL AUTHENTICATE-Variante aus 14.7.2.4 unterstützen.
- d. Das COS KANN weitere GENERAL AUTHENTICATE-Varianten
 1. unterstützen oder
 2. ablehnen.

(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 14.9.9.3)
 1. in *channelContext.keyReferenceList.internalAuthenticate*
 - i. eine Schlüsselreferenz eingetragen ist, die auf ein privates ELC-Schlüsselobjekt verweist und
 - ii. dort *algID* aus der Menge {*elcSessionkey4SM*, *elcSessionkey4TC*} ist und
 2. *channelContext.keyReferenceList.externalAuthenticate* leer ist.
- b. (Fall asynchrone, sym. Administration, Schlüsselauswahl gemäß 14.9.9.6)
 1. in *channelContext.keyReferenceList.externalAuthenticate* eine Schlüsselreferenz eingetragen ist und dort *algID* gleich *aesSessionkey4SM* ist und
 2. *channelContext.keyReferenceList.internalAuthenticate* leer ist.
- c. (Fall PACE, Schlüsselauswahl gemäß 14.9.9.7)

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 ist.
- d. (Fall asynchrone, asym. Administration, Schlüsselauswahl gemäß 14.9.9.3)
 1. in *channelContext.keyReferenceList.internalAuthenticate*
 - i. eine Schlüsselreferenz eingetragen ist, die auf ein privates ELC-Schlüsselobjekt verweist und
 - ii. dort *algID* aus der Menge {*elcAsynchronAdmin*} ist und
 2. *channelContext.keyReferenceList.externalAuthenticate* leer ist.

Hinweis (114): 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.

(N085.052) K_COS

Falls dies der erste Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist, dann werden folgende Schritte ausgeführt:

- a. Falls die acht LSByte von *keyRef* identisch sind zu *iccsn8* (siehe (N019.900)c), genau dann MUSS das Kommando mit dem Trailer AuthenticationFailure terminieren.
- b. Falls *channelContext.keyReferenceList.internalAuthenticate* nicht leer ist, dann wird *affectedObject_PrK* = SearchKey(
 channelContext.currentFolder,
 channelContext.keyReferenceList.internalAuthenticate.keyReference,
 channelContext.keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler

1. keyNotFound meldet, genau dann MUSS das Kommando mit dem Trailer PrKNotFound terminieren.
2. notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.
- c. Es wird *affectedObject_PuK* = SearchKey(
 channelContext.currentFolder,
 keyRef,
 channelContext.keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
- d. Falls *affectedObject_PuK.expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.
- e. Falls *affectedObject_PrK.domainParameter* verschieden ist von *affectedObject_PuK.domainParameter*, dann MUSS das Kommando mit dem Trailer ParameterMismatch terminieren.
- f. Zu den Domainparametern von *affectedObject_PrK* bzw. *affectedObject_PuK* wird ein ephemeres Schlüsselpaar (*ephemeralSK_self*, *ephemeralPK_self*) erzeugt.
- g. Der private Schlüssel *ephemeralSK_self* MUSS mindestens bis zum nächsten Schritt dieses Authentisierungsprotokolls gespeichert werden.
- h. Der öffentliche Schlüssel *ephemeralPK_self* MUSS DER codiert wie folgt in die Antwortdaten eingestellt werden:
responseData = ' 7C-L_{7C}-[85-L₈₅-P2OS(
 ephemeralPK_self,
 affectedObject_PrK.domainParameter.L
)]'.

(N085.054) K_COS

Falls dies der zweite Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist, dann werden die in (N085.052) ausgewählten Objekte *affectedObject_PrK* und *affectedObject_PuK* in den folgende Schritten verwendet:

- 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: Falls (N085.014)a eingehalten wird, ist diese Operation stets fehlerfrei.

- b. Falls AccessRuleEvaluation(*affectedObject_PrK*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.
- c. Das COS MUSS die Werte zur Ableitung der Sessionkeys wie folgt berechnen:

1. $K1 = \text{ECKA}(\text{affectedObject_PrK.d}, \text{PK_oponent}, \text{affectedObject_PrK.domainParameter})$.
 2. $K2 = \text{ECKA}(\text{ephemeralSK_self}, \text{affectedObject_PuK}, \text{affectedObject_PrK.domainParameter})$.
 3. $KD.i = K1 \parallel K2$
 4. $KD.e = \text{I2OS}(0, \text{OctetLength}(KD.i)) = '00...00'$.
 5. Die Oktettstrings $KD.e$ und $KD.i$ MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).
- d. Es MUSS `setSecurityStatus(affectedObject_PuK)` ausgeführt werden.
- e. Die Antwortdaten MÜSSEN leer sein: `responseData = ''`.

(N085.056) K_COS , Option_Kryptobox

Falls dies der zweite Schritt des Authentisierungsprotokolls für eine gegenseitige ELC-Authentisierung ist, dann werden die in (N085.052) ausgewählten Objekte `affectedObject_PrK` und `affectedObject_PuK` in den folgenden Schritten verwendet:

- a. Der ephemere öffentliche Schlüssel $PK_oponent$ des Protokollpartners wird wie folgt aus den Kommandodaten extrahiert:
- $$PK_oponent = \text{OS2P}(\text{ephemeralPK_oponent}, \text{affectedObject_PrK.domainParameter}).$$

Hinweis: Falls (N085.014)a eingehalten wird, ist diese Operation stets fehlerfrei.

- b. Falls `AccessRuleEvaluation(affectedObject_PrK, CLA, INS, P1, P2)` den Wert `False` zurückliefert, genau dann MUSS das Kommando mit dem Trailer `SecurityStatusNotSatisfied` terminieren.
- c. Das COS MUSS die Werte zur Ableitung der Sessionkeys wie folgt berechnen:
1. $K1 = \text{ECKA}(\text{ephemeralSK_self}, \text{affectedObject_PuK}, \text{affectedObject_PrK.domainParameter})$.
 2. $K2 = \text{ECKA}(\text{affectedObject_PrK.d}, \text{PK_oponent}, \text{affectedObject_PrK.domainParameter})$.
 3. $KD.i = K1 \parallel K2$
 4. $KD.e = \text{I2OS}(0, \text{OctetLength}(KD.i)) = '00...00'$.
 5. Die Oktettstrings $KD.e$ und $KD.i$ MÜSSEN an den Secure Messaging Layer übergeben werden (siehe 13.1).
- d. Falls das Kommando mit dem Trailer `NoError` antwortet, genau dann MUSS `setSecurityStatus(affectedObject_PuK)` ausgeführt werden.
- e. Die Antwortdaten MÜSSEN leer sein: `responseData = ''`.

(N085.060) K_COS

Falls dies der erste Schritt des Authentisierungsprotokolls für eine asynchrone, symmetrische Kartenadministration ist, dann werden folgende Schritte ausgeführt:

- a. Falls `channelContext.keyReferenceList.externalAuthenticate`
1. leer ist, genau dann MUSS das Kommando mit dem Trailer `NoKeyReference` terminieren.

2. nicht leer ist, dann wird *affectedObject* = SearchKey(
 channelContext.currentFolder,
 keyReferenceList.externalAuthenticate.keyReference,
 keyReferenceList.externalAuthenticate.algID
) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
 - b. Es gilt: *responseData* = '7C-04-(81-02-I2OS(*affectedObject.numberScenario*))'.
- (N085.062) K_COS
- Falls dies der zweite Schritt des Authentisierungsprotokolls für eine asynchrone, symmetrische Kartenadministration ist, dann wird das in (N085.060) ausgewählten Objekte *affectedObject* in den folgenden Schritten verwendet:
- a. Falls *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.
 - b. *cmdData* wird wie folgt aufgeteilt:
 1. *cmdData* = *M* || *MAC*.
 2. *OctetLength(MAC)* = 8.
 - c. *out* = *VerifyCMAC_IsoPadding(affectedObject.macKey, MAC, M)*
 - d. Falls *out* den Wert *INVALID* besitzt, dann MUSS das Kommando mit dem Trailer *AuthenticationFailure* terminieren.
 - e. Die Nachricht *M* wird wie folgt aufgeteilt:
 1. *M* = *NumberPrecondition* || *NumberScenario* || *C*.
 2. *OctetLength(NumberPrecondition)* = 2.
 3. *OctetLength(NumberScenario)* = 2.
 4. *OctetLength(C)* = 64.
 - f. Falls *affectedObject.numberScenario*
 1. kleiner als *OS2I(NumberPrecondition)* ist, genau dann MUSS das Kommando mit dem Trailer *NumberPreconditionWrong* terminieren.
 2. größer gleich *OS2I(NumberScenario)* ist, genau dann MUSS das Kommando mit dem Trailer *NumberScenarioWrong* terminieren.
 - g. *affectedObject.numberScenario* MUSS transaktionsgesichert auf den Wert *OS2I(NumberScenario)* gesetzt werden.
 - 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 13.1).

- i. Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS `setSecurityStatus(affectedObject)` ausgeführt werden.
- j. Die Antwortdaten MÜSSEN leer sein: `responseData = ''`.

(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 und dies ist

- a. der erste Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:

1. Schlüsselsuche `affectedObject = SearchKey(channelContext.currentFolder, channelContext.keyReferenceList.internalAuthenticate.keyReference, channelContext.keyReferenceList.internalAuthenticate.algID)` gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
2. Falls `AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)` den Wert `False` zurückliefert, genau dann MUSS das Kommando mit dem Trailer `SecurityStatusNotSatisfied` terminieren.
3. 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, \dots, z_n und MUSS in einen Oktettstring `CAN = 'o_1, ..., o_n'` umgewandelt werden, indem jede Ziffer z_i in ein Oktett o_i transformiert wird mit
 - i. $z_i = 0 \rightarrow o_i = '30'$,
 - ii. $z_i = 1 \rightarrow o_i = '31'$,
 - iii. $z_i = 2 \rightarrow o_i = '32'$,
 - iv. ...
 - v. $z_i = 9 \rightarrow o_i = '39'$,
4. $s = \text{RAND}(16)$,
5. $z = \text{AES_ENC}(\text{KDF}(\text{CAN}, 3, \text{channelContext.keyReferenceList.internalAuthenticate.algID}), s)$,
6. `responseData = '7C 12 (80 10 z)'`.

- b. der zweite Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:

1. Zuordnung `channelContext.keyReferenceList.internalAuthenticate.algID` zu `dP` (diese Zuordnung gilt auch für die übrigen Schritte des Protokolls):
 - i. id-PACE-ECDH-GM-AES-CBC-CMAC-128 $\rightarrow dP = \text{brainpoolP256r1}$
 - ii. id-PACE-ECDH-GM-AES-CBC-CMAC-192 $\rightarrow dP = \text{brainpoolP384r1}$

- iii. id-PACE-ECDH-GM-AES-CBC-CMAC-256 $\rightarrow dP = \text{brainpoolP512r1}$
- 2. Aus dem Wertfeld $value_{81}$ des DO81 im Kommandodatenfeld wird ein Punkt extrahiert: $\sim PK1_{PCD} = \text{OS2P}(value_{81}, dP)$,
- 3. Es wird ein ephemeres Schlüsselpaar generiert:
 - i. $\sim SK1_{P1CC} = \text{zufällige Zahl aus dem Intervall } [1, dP.n - 1]$,
 - ii. $\sim PK1_{P1CC} = \sim SK1_{P1CC} * dP.G$,
- 4. Berechne den ephemeren Basispunkt $\sim G$ mit s aus Schritt 1:
 $\sim G = s * dP.G + \sim SK1_{P1CC} * \sim PK1_{PCD}$,
- 5. Berechne ephemere Domainparameter $\sim D$ aus dP und $\sim G$:
 $\sim D = (dP.p, dP.a, dP.b, \sim G, dP.n, dP.h, dP.L, dP.\tau, dP.OID)$,
- 6. *responseData* MUSS ein DER codiertes DO7C wie folgt enthalten:
 $responseData = '7C - L_{7C} - (82 - L_{82} - P2OS(\sim PK1_{P1CC}, dP.\tau / 8))'$
- c. der dritte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
 - 1. Aus dem Wertfeld $value_{83}$ des DO83 im Kommandodatenfeld wird ein Punkt extrahiert: $\sim PK2_{PCD} = \text{OS2P}(value_{83}, \sim D)$,
 - 2. Es wird ein weiteres ephemeres Schlüsselpaar generiert:
 - i. $\sim SK2_{P1CC} = \text{zufällige Zahl aus dem Intervall } [1, \sim D.n - 1]$,
 - ii. $\sim PK2_{P1CC} = \sim SK2_{P1CC} * \sim D.G$,
 - 3. *responseData* MUSS ein DER codiertes DO7C wie folgt enthalten:
 $responseData = '7C - L_{7C} - (84 - L_{84} - P2OS(\sim PK2_{P1CC}, \sim D.\tau / 8))'$
- d. der vierte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
 - 1. Berechne gemeinsames Geheimnis zur Ableitung von Sessionkeys:
 - i. $KD.i = K_{AB} = \text{ECKA}(\sim SK2_{P1CC}, \sim PK2_{PCD}, \sim D)$,
 - ii. $KD.e = \text{I2OS}(0, \text{OctetLength}(KD.i)) = '00...00'$,
 - 2. Die *algID* aus *channelContext.keyReferenceList.internalAuthenticate* bestimmt, wie der Schlüssel k_{MAC} berechnet wird: $k_{MAC} = \text{KDF}(KD.i, 2, algID)$,
 - 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:
 $M_{PCD} = '7F49 - L_{7F49} - [(06 0A OID) || (86 - L_{86} - P2OS(\sim PK2_{P1CC}, \sim D))]'$,
 - 5. Berechne den DER codierten Oktettstring M_{P1CC} gemäß:
 $M_{P1CC} = '7F49 - L_{7F49} - [(06 0A OID) || (86 - L_{86} - P2OS(\sim PK2_{PCD}, \sim D))]'$,
 - 6. Überprüfe den MAC aus der Kommandonachricht: Falls
 $result = \text{VerifyCMAC_NoPadding}(k_{MAC}, T_{PCD}, M_{PCD})$ den Wert
 - i. INVALID besitzt, dann MUSS
 - A. *clearSecurityStatusKey(affectedObject)* ausgeführt werden und
 - B. das Kommando mit dem Trailer *AuthenticationFailure* terminieren.

ii. VALID besitzt, dann

- A. MUSS der MAC für die Antwortnachricht wie folgt berechnet werden:
 $T_{PICC} = \text{CalculateCMAC_NoPadding}(k_{MAC}, M_{PICC})$, 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 13.1), und
- D. MUSS das Datenfeld der Antwortnachricht wie folgt berechnet werden: $responseData = '7C\ 0A\ (86\ 08\ T_{PICC})'$.

(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 und dies ist

a. der erste Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:

1. Schlüsselsuche `affectedObject = SearchKey(channelContext.currentFolder, channelContext.keyReferenceList.internalAuthenticate.keyReference, channelContext.keyReferenceList.internalAuthenticate.algID)` gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
2. Falls `AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)` den Wert `False` zurückliefert, genau dann MUSS das Kommando mit dem Trailer `SecurityStatusNotSatisfied` terminieren.
3. Zuordnung `channelContext.keyReferenceList.internalAuthenticate.algID` zu `dP` (diese Zuordnung gilt auch für die übrigen Schritte des Protokolls).
 - i. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128 $\rightarrow dP = \text{brainpoolP256r1}$
 - ii. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192 $\rightarrow dP = \text{brainpoolP384r1}$
 - iii. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256 $\rightarrow dP = \text{brainpoolP512r1}$
4. Es wird ein ephemeres Schlüsselpaar generiert:
 - i. $\sim SK1_{PCD} = \text{zufällige Zahl aus dem Intervall } [1, dP.n - 1]$,
 - ii. $\sim PK1_{PCD} = \sim SK1_{PCD} * dP.G$,
5. `responseData` MUSS ein DER codiertes DO7C wie folgt enthalten:
 $responseData = '7C - L_{7C} - (81 - L_{81} - P2OS(\sim PK1_{PCD}, dP.\tau / 8))'$

b. der zweite Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:

1. Aus dem Datenfeld der Kommandonachricht werden z und `CAN` extrahiert.
2. $s = \text{OS2I}(\text{AES_DEC}(\text{KDF}(\text{CAN}, 3, \text{channelContext.keyReferenceList.internalAuthenticate.algID}),$

$\left. \begin{matrix} z \\ \end{matrix} \right\} \right\},$

3. $responseData = ''$ (leerer Oktettstring).
- c. der dritte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
 1. Aus dem Wertfeld $value_{82}$ des DO82 im Kommandodatenfeld wird ein Punkt extrahiert: $\sim PK1_{PICC} = OS2P(value_{82}, dP)$,
 2. Berechne den ephemeren Basispunkt $\sim G$ mit s aus Schritt 2:
 $\sim G = s * dP.G + \sim SK1_{PCD} * \sim PK1_{PICC}$,
 3. Berechne ephemere Domainparameter $\sim D$ aus dP und $\sim G$:
 $\sim D = (dP.p, dP.a, dP.b, \sim G, dP.n, dP.h, dP.L, dP.\tau, dP.OID)$,
 4. Es wird ein weiteres ephemeres Schlüsselpaar generiert:
 - i. $\sim SK2_{PCD} =$ zufällige Zahl aus dem Intervall $[1, \sim D.n - 1]$,
 - ii. $\sim PK2_{PCD} = \sim SK2_{PCD} * \sim D.G$,
 5. $responseData$ MUSS ein DER codiertes DO7C wie folgt enthalten:
 $responseData = '7C - L_{7C} - (83 - L_{83} - P2OS(\sim PK2_{PCD}, \sim D.\tau / 8))'$
- d. der vierte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
 1. Aus dem Wertfeld $value_{84}$ des DO84 im Kommandodatenfeld wird ein Punkt extrahiert: $\sim PK2_{PICC} = OS2P(value_{84}, \sim D)$,
 2. Berechne gemeinsames Geheimnis zur Ableitung von Sessionkeys:
 - i. $KD.i = K_{AB} = ECKA(\sim SK2_{PCD}, \sim PK2_{PICC}, \sim D)$,
 - ii. $KD.e = I2OS(0, OctetLength(KD.i)) = '00...00'$,
 3. Die $algID$ aus $channelContext.keyReferenceList.internalAuthenticate$ bestimmt, wie der Schlüssel k_{MAC} berechnet wird: $k_{MAC} = KDF(KD.i, 2, algID)$,
 4. Für den Zusammenhang zwischen $algID$ aus $externalAuthenticate$ oder $internalAuthenticate$ in $channelContext.keyReferenceList$ und OID gilt hier:
 - i. $algID = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128$
 $\Rightarrow OID = id-PACE-ECDH-GM-AES-CBC-CMAC-128$
 - ii. $algID = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192$
 $\Rightarrow OID = id-PACE-ECDH-GM-AES-CBC-CMAC-192$
 - iii. $algID = id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256$
 $\Rightarrow OID = id-PACE-ECDH-GM-AES-CBC-CMAC-128$
 5. Berechne den Oktettstring M_{PCD} identisch zu (N085.064)d.4 und den zugehörigen MAC: $T_{PCD} = CalculateCMAC_NoPadding(k_{MAC}, M_{PCD})$.
 6. Berechne den Oktettstring M_{PICC} identisch zu (N085.064)d.5.
 7. Setze $responseData = '7C 0A (85 08 T_{PCD})'$
- e. der fünfte Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
 1. Überprüfe den MAC aus der Kommandonachricht: Falls
 $result = VerifyCMAC_NoPadding(k_{MAC}, T_{PICC}, M_{PICC})$ den Wert

- i. INVALID besitzt, dann MUSS
 - A. `clearSecurityStatusKey(affectedObject)` ausgeführt werden und
 - B. das Kommando mit dem Trailer `AuthenticationFailure` terminieren.
- ii. 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 13.1), und
 - C. MUSS gelten `responseData = ''` (leerer Oktettstring).

(N085.068) K_COS

Asynchrone, asymmetrische Kartenadministration:

Falls `channelContext.keyReferenceList.internalAuthenticate.algID` der Algorithmus `elcAsynchronAdmin` eingetragen ist und dies ist

- a. der erste Schritt des Authentisierungsprotokolls, dann werden folgende Schritte ausgeführt:
 - 1. Schlüsselsuche `affectedObject_PrK = SearchKey(`
`channelContext.currentFolder,`
`channelContext.keyReferenceList.internalAuthenticate.keyReference,`
`channelContext.keyReferenceList.internalAuthenticate.algID`
`)` gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
 - 2. Es wird `affectedObject_PuK = SearchKey(`
`channelContext.currentFolder,`
`keyRef,`
`channelContext.keyReferenceList.internalAuthenticate.algID`
`)` gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
 - 3. Falls `affectedObject_PuK.expirationDate` kleiner als `pointInTime` ist (siehe (N019.900)), dann MUSS das Kommando mit dem Trailer `KeyExpired` terminieren.
 - 4. Falls `affectedObject_PrK.domainParameter` verschieden ist von `affectedObject_PuK.domainParameter`, dann MUSS das Kommando mit dem Trailer `ParameterMismatch` terminieren.
 - 5. Es gilt:
`responseData = '7C-04-(81-02-l2OS(affectedObject_PrK.numberScenario))'`.
- b. der zweite Schritt des Authentisierungsprotokolls ist, dann werden folgende Schritte ausgeführt:

1. Falls `AccessRuleEvaluation(affectedObject_PrK, CLA, INS, P1, P2)` den Wert `False` zurückliefert, genau dann MUSS das Kommando mit dem Trailer `SecurityStatusNotSatisfied` terminieren.
2. `cmdData` wird wie folgt aufgeteilt:
 - i. $cmdData = M \parallel SIG$.
 - 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. $SIG = R \parallel S$
 - iv. $OctetLength(R) = OctetLength(S)$.
3. Falls `ELC_VER_SIG(affectedObject_PuK, R, S, hash)` den Wert `False` besitzt, MUSS das Kommando mit dem Trailer `AuthenticationFailure` terminieren.
4. Die Nachricht `M` wird wie folgt aufgeteilt:
 - i. $M = NumberPrecondition \parallel NumberScenario \parallel cipher$.
 - ii. $OctetLength(NumberPrecondition) = 2$.
 - iii. $OctetLength(NumberScenario) = 2$.
5. Falls `affectedObject_PrK.numberScenario`
 - i. kleiner als `OS2I(NumberPrecondition)` ist, genau dann MUSS das Kommando mit dem Trailer `NumberPreconditionWrong` terminieren.
 - ii. größer gleich `OS2I(NumberScenario)` ist, genau dann MUSS das Kommando mit dem Trailer `NumberScenarioWrong` terminieren.
6. `affectedObject_PrK.numberScenario` MUSS transaktionsgesichert auf den Wert `OS2I(NumberScenario)` gesetzt werden.
7. Es gilt (*Hinweis: 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. $cipher = 'A6-L_{A6}-(oidDO \parallel keyDO \parallel cipherDO \parallel macDO)$.
 - iii. $oidDO = '06-L_{06}-oid'$.
 - iv. $keyDO = '7F49-L_{7F49}-(86-L_{86}-PO_A)'$.
 - v. $cipherDO = '86-L_{86}-(02 \parallel C)'$.
 - 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 13.1).

8. Falls das Kommando mit dem Trailer NoError antwortet, genau dann MUSS `setSecurityStatus(affectedObject_PuK)` ausgeführt werden.

9. Die Antwortdaten MÜSSEN leer sein: `responseData = ""`.

Hinweis (115): Wegen (N008.600)d und Tabelle 271 ist L_{06} in (N085.068)b.7.iii stets 9 und damit `oidDO` stets 11 Oktett lang.

Hinweis (116): 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: $L_{86} = 65 \Rightarrow \text{keyDO}$ ist 70 Oktett lang.*
- b. *brainpoolP384r1: $L_{86} = 97 \Rightarrow \text{keyDO}$ ist 102 Oktett lang.*
- c. *brainpoolP512r1: $L_{86} = 129 \Rightarrow \text{keyDO}$ ist 136 Oktett lang.*

Hinweis (117): 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 (118): Wegen (N002.810)h ist L_{8E} in (N085.068)b.7.vi gleich 8 und damit `macDO` 10 Oktett lang.

(N085.070) K_COS

Als Datenfeld der Antwortnachricht MUSS der (möglicherweise leere) Oktettstring `responseData` verwendet werden.

(N085.072) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N085.074) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 174 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 174 MUSS eine höhere Priorität als NoError haben.

Hinweis (119): Im Rahmen dieser Spezifikation sind Authentisierungssequenzen nicht unterbrechbar (siehe (N104.600)).

Hinweis (120): 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 (121): Dieses Kommando ist nicht in der Normenreihe ISO/IEC 7816 enthalten. Es ließe sich kombinieren mit dem Kommando EXTERNAL AUTHENTICATE.

Hinweis (122): Bei der Spezifikation der Kommando-APDU wurde das MSE-Set-Kommando zugrunde gelegt.

Hinweis (123): 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

In dieser Variante enthält die APDU des MSE Kommandos einen Parameter:

(N085.100) K_externeWelt {K_Karte}, Option_DES

Der Parameter *cmdData* enthält eine Schlüsselreferenz. Wert und Codierung MÜSSEN gemäß (N099.600) gewählt werden.

(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 Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 175 verwendet werden.

Tabelle 175: 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 <i>cmdData</i> '

14.7.3.2 Use Case Auslesen des Sicherheitsstatus einer Rolle, Option_RSA_CVC

In dieser Variante enthält die APDU des MSE Kommandos einen Parameter:

(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.

(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 Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 176 verwendet werden.

Tabelle 176: 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 - <i>cmdData</i> '

14.7.3.3 Use Case Auslesen des Sicherheitsstatus einer Bitliste

In dieser Variante enthält die APDU des MSE Kommandos zwei Parameter:

(N085.440) K_externeWelt {K_Karte}

Der Parameter *oid* MUSS eine OID aus der Menge {oid_cvc_fl_ti, oid_cvc_fl_cms} enthalten.

(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.

(N085.444) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 177 verwendet werden.

Tabelle 177: 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 – <i>oid</i> 53 - 07 - <i>cmdData</i>)'

14.7.3.4 Antwort der Karte auf Auslesen Sicherheitsstatus eines Schlüssels

Tabelle 178: 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 179: 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 (124): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N085.500) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.7.3.5 Kommandoabarbeitung innerhalb der Karte

(N085.600) K_COS

- a. Das COS MUSS die GET SECURITY STATUS KEY-Varianten aus
 1. Option_DES, 14.7.3.1,
 2. Option_RSA_CVC, 14.7.3.2 und
 3. 14.7.3.3 unterstützen.

- b. Das COS KANN weitere GET SECURITY STATUS KEY-Varianten
 - 1. unterstützen oder
 - 2. ablehnen.

(N085.700) K_COS, Option_DES

Falls *cmdData* die Referenz eines symmetrischen Schlüssels gemäß 14.7.3.1 ist, dann

- a. wird *affectedObject* = SearchSecretKey(
 channelContext.currentFolder,
 cmdData,
 Wildcard
) gesetzt. Gemäß 9.2.3 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: Der Fehler notSupported ist wegen der Wildcard-Suche nicht möglich.
- b. Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.
- c. Falls *affectedObject* in *globalSecurityList* (siehe (N029.900)e) oder in *dfSpecificSecurityList* (siehe (N029.900)f) enthalten ist, dann MUSS als Trailer NoError verwendet werden.

(N085.800) K_COS, Option_RSA_CVC

Falls *cmdData* eine Rolle gemäß 14.7.3.2. 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.

(N085.900) K_COS

Falls *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 Hinweis (51):), dann MUSS als Trailer NoError verwendet werden.

(N086.000) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoAuthentication gewählt werden.

(N086.100) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 179 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 179 MUSS eine höhere Priorität als NoError haben.
- c. 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 14.9.9.3). Das Token ist als Parameter in der Kommandonachricht enthalten.

14.7.4.1 Use Case interne Authentisierung

In dieser Variante enthält die APDU des INTERNAL AUTHENTICATE-Kommandos zwei Parameter:

(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* ist abhängig von der mittels (N100.800) ausgewählten *algld*. Wenn *algld* gleich

- elcRoleAuthentication ist, dann MUSS *token* gleich 24 Oktett lang sein.
- rsaClientAuthentication ist, dann DARF *token* NICHT länger als 64 Oktette sein.
- Option_RSA_CVC, rsaRoleAuthentication ist, dann MUSS *token* gleich 16 Oktette lang sein.
- Option_DES, rsaSessionkey4SM ist, dann MUSS *token* gemäß (N106.700) gewählt werden.
- signPKCS1_V1_5 ist, dann MUSS die Anzahl Oktette in *token* kleiner als $0,4 \cdot \text{OktetLength}(n)$ sein, mit n als Modulus des ausgewählten Authentisierungsschlüssels.

(N086.202) K_externeWelt {K_Karte}, Option_Kryptobox

Der Parameter *token* enthält die zu authentisierenden Daten. Der Parameter *token* ist ein Oktettstring mit beliebigem Inhalt. Die Länge von *token* ist abhängig von der mittels (N100.800) ausgewählten *algld*. Wenn *algld* gleich

- aesSessionkey4TC ist, dann MUSS *token* gleich 24 Oktett lang sein.
- Option_DES, desSessionkey4TC ist, dann MUSS *token* gleich 16 Oktett lang sein.
- Option_DES, rsaSessionkey4TC ist, dann MUSS *token* gleich 16 Oktett lang sein.

(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.

(N086.400) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 180 verwendet werden.

Tabelle 180: 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'	<i>token</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.7.4.2 Antwort der Karte auf interne Authentisierung

Tabelle 181: INTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>response</i>	Authentisierende Daten
Trailer	Inhalt	Beschreibung
'90 00'	NoError	Erfolgreiche Authentisierungsoperation

Tabelle 182: 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 angegebenen Algorithmus nicht
'6A 88'	KeyNotFound	Authentisierungsschlüssel nicht gefunden
'6A 88'	PukNotFound	Verschlüsselungsschlüssel nicht gefunden

Hinweis (125): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N086.500) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.7.4.3 Kommandoabarbeitung innerhalb der Karte

(N086.600) K_COS

- Das COS MUSS die INTERNAL AUTHENTICATE-Variante aus 14.7.4.1 unterstützen.
- Das COS KANN weitere INTERNAL AUTHENTICATE-Varianten
 - unterstützen oder
 - ablehnen.

(N086.700) K_COS

Wenn *channelContext.keyReferenceList.internalAuthenticate*

- leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
- nicht leer ist, dann wird *affectedObject* = SearchKey(
 currentFolder,
 keyReferenceList.internalAuthenticate.keyReference,
 keyReferenceList.internalAuthenticate.algID
) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.

(N086.800) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.

(N086.810) K_COS

Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer *KeyInvalid* terminieren.

(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.

(N086.900) K_COS

Die Antwort *response* wird wie folgt berechnet:

Wenn *channelContext.keyReferenceList.internalAuthenticate.algID* den Wert

- a. *elcRoleAuthentication* besitzt, dann gilt mit $PrK = affectedObject.privateKey$

$$response = ELC_SIG(\begin{array}{l} PrK, \\ I2OS(OS2I(token \parallel elcRoleAuthentication), \\ PrK.domainParameter.\tau / 8 \end{array}))$$

- b. *rsaClientAuthentication* besitzt, dann gilt

$$response = RSASSA_PSS_SIGN(affectedObject.privateKey, token)$$

- c. *Option_RSA_CVC*, *rsaRoleAuthentication* besitzt, dann gilt mit $PrK = affectedObject.privateKey$

1. $PRND = RAND(OctetLength(PrK.n) - 34)$

2. $M = PRND \parallel token$

3. $(response, M2) = RSA_ISO9796_2_DS1_SIGN(PrK, M)$

Hinweis: Die Länge von *PRND* ist so gewählt, dass *M2 = token* ist. Darum ist es nicht notwendig, *M2* in die Antwortnachricht einzustellen.

- d. *Option_DES*, *rsaSessionkey4SM* besitzt, dann gilt mit $PrK = affectedObject.privateKey$

1. $KD.i = RAND(64)$

Der Oktettstring *KD.i* MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).

2. $PRND = RAND(OctetLength(PrK.n) - 34 - OctetLength(KD.i))$

3. $M = PRND \parallel KD.i \parallel token$

4. $(sig, M2) = \text{RSA_ISO9796_2_DS1_SIGN}(PrK, M)$
Hinweis: Die Länge von PRND ist so gewählt, dass $M2 = \text{token}$ ist. Darum ist es nicht notwendig, $M2$ in die Antwortnachricht einzustellen.
5. Wenn *channelContext.keyReferenceList.externalAuthenticate*
- i. leer ist, genau dann MUSS das Kommando mit dem Trailer NoPukReference terminieren.
 - ii. nicht leer ist, dann wird $tmpObject = \text{SearchKey}(currentFolder, keyReferenceList.externalAuthenticate.keyReference, keyReferenceList.internalAuthenticate.algID)$ gesetzt. Gemäß 9.2.3.2 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
6. $response = sig^{PuK.e} \bmod PuK.n$, mit $PuK = tmpObject.publicKey$.
- e. signPKCS1_V1_5 besitzt, dann gilt:
 $response = \text{RSASSA_PKCS1_V1_5_SIGN}(PrK, token)$
- (N086.902) K_COS, Option_Kryptobox
Die Antwort *response* wird wie folgt berechnet:
Wenn *channelContext.keyReferenceList.internalAuthenticate.algID* den Wert
- a. aesSessionkey4TC besitzt, dann werden folgende Schritte ausgeführt:
 - 1. Setze $RND.int = \text{RAND}(16)$
 - 2. Setze $ICCSN8.int = iccsn8$ (siehe (N019.900)c)
 - 3. Setze $KD.i = \text{RAND}(64)$
 - 4. Setze $S = RND.int \parallel ICCSN8.int \parallel token \parallel KD.i$
 - 5. Setze $C1 = \text{AES_CBC_ENC}(affectedObject.encKey, 0, S)$
 - 6. Setze $MAC1 = \text{CalculateCMAC_IsoPadding}(affectedObject.macKey, C1)$
 - 7. Setze $response = C1 \parallel MAC1$
 - 8. $RND.int$ MUSS als $RND.ICC$ gespeichert werden (siehe (N029.900)b).
 - 9. $KD.i$ MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).
 - b. Option_DES, desSessionkey4TC besitzt, dann werden folgende Schritte ausgeführt:
 - 1. Setze $RND.int = \text{RAND}(8)$
 - 2. Setze $ICCSN8.int = iccsn8$ (siehe (N019.900)c)
 - 3. Setze $KD.i = \text{RAND}(64)$
 - 4. Setze $S = RND.int \parallel ICCSN8.int \parallel token \parallel KD.i$
 - 5. Setze $C1 = \text{3DES_CBC_ENC}(affectedObject.encKey, 0, S)$
 - 6. Setze $MAC1 = \text{CALCULATE_Retail_MAC}(affectedObject.macKey, C1)$
 - 7. Setze $response = C1 \parallel MAC1$

- 8. RND.int MUSS als *RND.ICC* gespeichert werden (siehe (N029.900)b).
- 9. KD.i MUSS an den Secure Messaging Layer übergeben werden (siehe 13.1).
- c. Option_DES, rsaSessionkey4TC besitzt, dann MUSS im Rahmen der Kommandobearbeitung identisch zu rsaSessionkey4SM verfahren werden.

(N087.000) K_COS

Als Datenfeld der Antwortnachricht MUSS *response* verwendet werden.

(N087.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N087.200) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 182 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 182 MUSS eine höhere Priorität als NoError haben.

Hinweis (126): 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 Kryptoboxkommandos

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 183: Tabelle aller PSO 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, siehe [ISO/IEC 7816-8]	-
		'86'	'B8'	PSO Transcipher mittels RSA	14.8.6.1
		'86'	'B8'	PSO Transcipher mittels ELC	14.8.6.3
		'00'	'AE'	PSO Verify Certificate	14.8.7
		'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, 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)}.

(N087.220) K_externeWelt {K_Karte}

In dieser Variante enthält die APDU des PSO Compute Cryptographic Checksum Kommandos drei Parameter:

- a. Der Parameter *flagSSCmacIncrement* MUSS wie folgt gewählt werden: Falls es sich um Sessionkeys für den Algorithmus
 1. Option_DES, desSessionkey handelt, dann ist *flagSSCmacIncrement* ein leerer Oktettstring.
 2. aesSessionkey handelt und SSCmac
 - i. zu inkrementieren ist, dann ist *flagSSCmacIncrement* = '01'.
 - ii. unverändert zu verwenden ist, dann ist *flagSSCmacIncrement* = '00'.
- b. 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.
- c. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {WildcardShort, WildCardExtended} gewählt werden.

(N087.228) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 184 verwendet werden.

Tabelle 184: 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'	<i>flagSSCmacIncrement</i> <i>data</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.8.1.2 Antwort der Karte auf Berechnen einer kryptographischen Checksumme

Tabelle 185: 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 186: 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 (127): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N087.232) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.1.3 Kommandoabarbeitung innerhalb der Karte

(N087.236) K_COS, Option_Kryptobox, Option_PACE_PCD

- Das COS MUSS die PSO Compute Cryptographic Checksum Variante aus 14.8.1.1 unterstützen.
- Das COS KANN weitere PSO Compute Cryptographic Checksum Varianten
 - unterstützen oder
 - ablehnen.

(N087.240) K_COS

Wenn *channelContext.keyReferenceList.macCalculation*

- leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
- nicht leer ist, dann wird *affectedObject* = SearchKey(
 currentFolder,
 keyReferenceList.macCalculation.keyReference,
 keyReferenceList.macCalculation.algID
) gesetzt. Gemäß 9.2.3 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.
 - notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.

(N087.244) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False

zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N087.248) K_COS

Mit den Attributen *SSCmac* und *Kmac* aus dem Attribut *SessionkeyContext* gilt:
Falls *keyReferenceList.macCalculation.algID* den Wert

a. *aesSessionkey* besitzt, MUSS gelten:

1. Falls *flagSSCmacIncrement* = '01' ist, dann wird *SSCmac* inkrementiert:
 $SSCmac = SSCmac + 1$, sonst bleibt *SSCmac* unverändert.
2. *mac* = CalculateCMAC_IsoPadding(*Kmac*, I2OS(*SSCmac*, 16) || *data*).

b. Option_DES, desSessionkey besitzt, MUSS gelten:

1. *SSCmac* = *SSCmac* + 1.
2. *mac* = CALCULATE_Retail_MAC(*Kmac*, I2OS(*SSCmac*, 8) || *data*).

(N087.252) K_COS

Als Datenfeld der Antwortnachricht MUSS *mac* verwendet werden.

(N087.256) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N087.260) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 186 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 186 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 14.9.9.9). 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:

(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* ist abhängig von der mittels (N102.800) ausgewählten *algID*. Wenn *algID* gleich

- a. *rsaClientAuthentication* ist, dann DARF *dataToBeSigned* nicht länger als 64 Oktette sein.
- b. *signECDSA* ist, dann MUSS die Länge von *dataToBeSigned* gleich dem Parameter *domainParameter.τ* des Signierschlüssels sein.

c. signPKCS1_V1_5 ist, dann MUSS die Anzahl Oktette in *dataToBeSigned* kleiner als $0,4 \cdot \text{OctetLength}(n)$ sein, mit n als Modulus des ausgewählten Signaturschlüssels.

d. signPSS ist, dann DARF *dataToBeSigned* nicht länger als 64 Oktette sein.

(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.

(N087.500) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 187 verwendet werden.

Tabelle 187: 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'	<i>dataToBeSigned</i>
Le	<i>length</i>	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:

(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 (128): Die Kapazität eines RSA-Schlüssels (siehe (N002.100)) beträgt für die Modulslänge

a. 2048 bit: $c = k - L_h - L_s - 8t - 2 = (2048 - 256 - 256 - 8 \cdot 1 - 2) \text{ bit} = 1526 \text{ bit}$.

b. 3072 bit: $c = k - L_h - L_s - 8t - 2 = (3072 - 256 - 256 - 8 \cdot 1 - 2) \text{ bit} = 2550 \text{ bit}$.

(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.

(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.

(N087.900) K_externeWelt {K_Karte}

Die Parameter *M1* und *hashM2* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N088.600)a spezifiziert. Die Anzahl der Oktette in *signInput9796_2_DS2* wird durch (N087.600) und (N087.700) beschränkt.

(N088.000) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 188 verwendet werden.

Tabelle 188: 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’	<i>signInput9796_2_DS2</i> , DER codiertes Datenfeld
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.8.2.3 Antwort der Karte auf Signieren von Daten

Tabelle 189: PSO Compute Digital Signature Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
‘xx...xx’	<i>signature</i>	Signatur
Trailer	Inhalt	Beschreibung
‘90 00’	NoError	Erfolgreiche Signaturoperation

Tabelle 190: 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 (129): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N088.100) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.2.4 Kommandoabarbeitung innerhalb der Karte

(N088.200) K_COS

- Das COS MUSS die PSO Compute Digital Signature-Varianten aus 14.8.2.1 und 14.8.2.2 unterstützen.

- b. Das COS KANN weitere PSO Compute Digital Signature-Varianten
 - 1. unterstützen oder
 - 2. ablehnen.

(N088.300) K_COS

Wenn *channelContext.keyReferenceList.signatureCreation*

- a. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
- b. nicht leer ist, dann wird *affectedObject = SearchSecretKey(*
currentFolder,
keyReferenceList.signatureCreation.keyReference,
keyReferenceList.signatureCreation.algID
) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.

(N088.400) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N088.500) K_COS

Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.

(N088.600) K_COS

Die Signatur *signature* MUSS wie folgt berechnet werden:

Wenn *keyReferenceList.signatureCreation.algID* den Wert

- a. *rsaClientAuthentication* oder *signPSS* besitzt, dann gilt
signature = RSASSA_PSS_SIGN(
affectedObject.privateRsaKey,
dataToBeSigned
)
- b. *sign9796_2_DS2* besitzt, dann gilt
 - 1. *signInput9796_2_DS2 = plainDO || hashDO.*
 - 2. *plainDO = '80 - L₈₀ - M1'.*
 - 3. *hashDO = '90 - L₉₀ - hashM2'.*
 - 4. *signature = RSA_ISO9796_2_DS2_SIGN(*
affectedObject.privateRsaKey,
M1,
hashM2,
)
- c. *signECDSA* besitzt, dann gilt
signature = R || S, mit
(R, S) = ELC_SIG(affectedObject.privateElcKey, dataToBeSigned)

d. signPKCS1_V1_5 besitzt, dann gilt:
signature = RSASSA_PKCS1_V1_5_SIGN(
 affectedObject.privateRsaKey,
 dataToBeSigned
)

(N088.700) K_COS

Als Datenfeld der Antwortnachricht MUSS *signature* verwendet werden.

(N088.800) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N088.900) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 190 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 190 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, 15.4.4) ausgehandelt und zusammen mit einem Algorithmus implizit ausgewählt.
- Wird ein asymmetrischer 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 14.9.9.11).

14.8.3.1 Use Case Entschlüsseln mittels RSA

Diese Variante gilt für Algorithmen (siehe (N017.900)) aus folgender Menge: {rsaDecipherPKCS1_V1_5, rsaDecipherOaep}.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos zwei Parameter:

(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.

(N089.100) K_externeWelt {K_Karte}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.

(N089.200) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 191 verwendet werden.

Tabelle 191: 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 Chiffre
Data	'XX...XX'	'00' C, dies bedeutet: PaddingIndicator Cryptogram
Le	'0000'	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten

14.8.3.2 Use Case Entschlüsseln mittels ELC

Diese Variante gilt für folgenden Algorithmus (siehe (N017.900)): `elcSharedSecretCalculation`.

In dieser Variante enthält die APDU des PSO Decipher-Kommandos vier Parameter:

(N089.300) `K_externerWelt {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).

(N089.400) `K_externerWelt {K_Karte}`

Der Parameter C enthält die zu entschlüsselnden Daten. Der Parameter C ist ein Oktettstring mit beliebigem Inhalt.

(N089.500) `K_externerWelt {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.

(N089.600) `K_externerWelt {K_Karte}`

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS aus der Menge {WildCardShort, WildCardExtended} gewählt werden.

(N089.700) `K_externerWelt {K_Karte}`

Die Parameter PO_A , C und T MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N090.300)c spezifiziert.

(N089.800) `K_externerWelt {K_Karte}`

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 192 verwendet werden.

Tabelle 192: 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 Chiffre

Data	'XX...XX'	<i>cipher</i> , DER codiertes Datenfeld
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

Hinweis (130): 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:

(N089.840) 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 ein ganzzahliges Vielfaches der Blocklänge des verwendeten Kryptoalgorithmus sein.

(N089.843) 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.

(N089.845) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 193 verwendet werden.

Tabelle 193: 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 Chiffre
Data	'XX...XX'	'01' <i>C</i> = Paddingindikator Kryptogramm
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.8.3.4 Antwort der Karte auf Entschlüsseln von Daten

Tabelle 194: PSO Decipher Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>plain</i>	Klartext
Trailer	Inhalt	Beschreibung
'90 00'	NoError	Erfolgreiche Entschlüsselungsoperation

Tabelle 195: 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 angegebenen Algorithmus nicht
'6A 88'	KeyNotFound	Schlüssel nicht gefunden

Hinweis (131): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N089.900) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.3.5 Kommandoabarbeitung innerhalb der Karte

(N090.000) Kommandovarianten

a. K_COS

Das COS MUSS die PSO Decipher-Varianten aus 14.8.3.1 und 14.8.3.2 unterstützen.

b. K_COS, Option_Kryptobox, Option_PACE_PCD

Das COS MUSS die PSO Decipher-Variante aus 14.8.3.3 unterstützen.

c. Das COS KANN weitere PSO Decipher-Varianten

1. unterstützen oder

2. ablehnen.

(N090.100) K_COS

Wenn *channelContext.keyReferenceList.dataDecipher*

a. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.

b. nicht leer ist, dann wird *affectedObject = SearchKey(*
currentFolder,
keyReferenceList.dataDecipher.keyReference,
keyReferenceList.dataDecipher.algID

) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.

(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.

(N090.210) K_COS

Wenn *affectedObject.keyAvailable* den Wert False besitzt, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.

(N090.300) K_COS

Die Klartextnachricht *plain* MUSS wie folgt berechnet werden:

Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert

- a. rsaDecipherPKCS1_V1_5 besitzt, dann gilt
 $plain = \text{RSAES_PKCS1_V1_5_DECRYPT}(affectedObject.privateRsaKey, C)$
- b. rsaDecipherOaep besitzt, dann gilt
 $plain = \text{RSAES_OAEP_DECRYPT}(affectedObject.privateRsaKey, C)$
- c. elcSharedSecretCalculation besitzt, dann gilt (*Hinweis: cipher ist hier identisch zu (N085.068)b.7, (N091.700)d und (N094.400)c definiert*)
 1. $cipher = 'A6-L_{A6}-(oidDO \parallel keyDO \parallel cipherDO \parallel macDO)$.
 2. $oidDO = '06-L_{06}-oid'$.
 3. $keyDO = '7F49-L_{7F49}-(86 - L_{86} - PO_A)'$.
 4. $cipherDO = '86-L_{86}-(02 \parallel C)'$.
 5. $macDO = '8E-L_{8E}-T'$.
 6. Falls *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. $plain = \text{ELC_DEC}(\text{OS2P}(PO_A, dP), affectedObject.privateElcKey, C, T)$.

(N090.302) K_COS, Option_Kryptobox

Die Klartextnachricht *plain* MUSS wie folgt berechnet werden:

Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert

- a. aesSessionkey besitzt, dann gilt mit den Attributen *SSCmac* und *Kenc* aus dem Attribut *SessionkeyContext*
 1. Schritt 1: $SSCenc = \text{OS2I}(\text{AES_ENC}(K_{enc}, \text{I2OS}(SSCmac, 16)))$.
 2. Schritt 2: $P = \text{AES_CBC_DEC}(K_{enc}, SSCenc, C)$.
 3. Schritt 3: $plain = \text{TruncateIso}(P, 16)$.
 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.
- b. Option_DES, desSessionkey besitzt, dann gilt mit den Attributen *SSCenc* und *Kenc* aus dem Attribut *SessionkeyContext*
 1. Schritt 1: $SSCenc = SSCenc + 1$.
 2. Schritt 2: $P = \text{3TDES_CBC_DEC}(K_{enc}, SSCenc, C)$.
 3. Schritt 3: $plain = \text{TruncateIso}(P, 8)$.
 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.

(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.

(N090.500) K_COS

Als Datenfeld der Antwortnachricht MUSS *plain* verwendet werden.

(N090.600) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N090.700) K_COS

Für die Priorität der Trailer gilt:

- Die Priorität der Trailer in Tabelle 195 ist herstellerspezifisch.
- Jeder Trailer in Tabelle 195 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, 15.4.4) ausgehandelt und zusammen mit einem Verschlüsselungsalgorithmus implizit ausgewählt.
- Wird ein asymmetrischer 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 14.9.9.12).
- Wird ein asymmetrischer 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, rsaEncipherPKCS1_V1_5}.

Hinweis (132): 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:

(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, rsaEncipherPKCS1_V1_5} gewählt werden.

(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){b.3, c.3}).

(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*. Wenn *algID* gleich

- rsaEncipherOaep ist, dann DARF $\text{OctetLength}(M)$ NICHT größer als $\text{OctetLength}(n) - 66$ sein.

b. `rsaEncipherPKCS1_V1_5` ist, dann DARF `OctetLength(M)` NICHT größer als `OctetLength(n) - 11` sein.

(N090.786) `K_externeWelt {K_Karte}`

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich `WildcardExtended` sein.

(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){b, c} spezifiziert.

(N090.790) `K_externeWelt {K_Karte}`

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 196 verwendet werden.

Tabelle 196: 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 Chiffre
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX...XX'	<i>plainDO</i> , DER codiertes Datenfeld
Le	'0000'	<i>length</i> , 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 (133): 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:

(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.

(N090.900) `K_externeWelt {K_Karte}`

Der Parameter *oid* enthält einen Objektidentifizierer, der die zu verwendende elliptische Kurve referenziert. Der Parameter *oid* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.

(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).

(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*.

(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.

(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.

(N091.400) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 197 verwendet werden.

Tabelle 197: 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 Chiffre
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX...XX'	<i>plainDO</i>
Le	<i>length</i>	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, rsaEncipherPKCS1_V1_5}.

Hinweis (134): 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:

(N091.420) K_externeWelt {K_COS}

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*. Wenn *algID* gleich

- rsaEncipherOaep ist, dann DARF OctetLength(*M*) NICHT größer als OctetLength(*n*) – 66 sein.
- rsaEncipherPKCS1_V1_5 ist, dann DARF OctetLength(*M*) NICHT größer als OctetLength(*n*) – 11 sein.

(N091.422) K_externeWelt {K_COS}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.

(N091.424) K_externeWelt {K_COS}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 198 verwendet werden.

Tabelle 198: 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 Chiffre
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX...XX'	<i>M</i>
Le	'0000'	<i>length</i> , 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 (135): 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:

(N091.430) K_externeWelt {K_COS}

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*.

(N091.432) K_externeWelt {K_COS}

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.

(N091.434) K_externeWelt {K_COS}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 199 verwendet werden.

Tabelle 199: 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 Chiffre
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX...XX'	<i>M</i>
Le	<i>length</i>	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 (136): 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:

(N091.440) K_externeWelt {K_COS}

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*.

(N091.443) K_externeWelt {K_COS}

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.

(N091.446) K_externeWelt {K_COS}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 200 verwendet werden.

Tabelle 200: 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 Chiffre
P2	'80'	Beschreibung der Kommandodaten, hier Klartext
Data	'XX...XX'	<i>M</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.8.4.6 Antwort der Karte auf Verschlüsseln von Daten

Tabelle 201: PSO Encipher Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>cipher</i>	Chiffre
Trailer	Inhalt	Beschreibung
'90 00'	NoError	Erfolgreiche Verschlüsselungsoperation

Tabelle 202: 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 angegebenen Algorithmus nicht
'6A 88'	KeyNotFound	Schlüssel nicht gefunden

Hinweis (137): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N091.500) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.4.7 Kommandoabarbeitung innerhalb der Karte

(N091.600) Kommandovarianten

- a. K_COS
Das COS MUSS die PSO Encipher-Variante aus 14.8.4.1 und 14.8.4.2 unterstützen.
- b. K_COS, Option_Kryptobox
Das COS MUSS die PSO Encipher-Variante aus 14.8.4.3 und 14.8.4.4 unterstützen.
- c. K_COS, Option_Kryptobox, Option_PACE_PCD
Das COS MUSS die PSO Encipher-Variante aus 14.8.4.5 unterstützen.
- d. Das COS KANN weitere PSO Encipher-Varianten
 1. unterstützen oder
 2. ablehnen.

(N091.650) K_COS

Wenn *channelContext.keyReferenceList.dataEncipher* nicht leer ist, genau dann MUSS das Kommando auf folgende Art mit einem in der Smartcard gespeicherten Schlüssel arbeiten:

- a. Es wird *affectedObject* = SearchKey(
 currentFolder,
 keyReferenceList.dataEncipher.keyReference,
 keyReferenceList.dataEncipher.algID
) gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.
- b. Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.
- c. Das Chiffre *cipher* MUSS wie folgt berechnet werden: Wenn *channelContext.keyReferenceList.dataEncipher.algID* den Wert
 1. aesSessionkey besitzt, dann gilt mit den Attributen *SSCmac* und *Kenc* aus dem Attribut *SessionkeyContext*
 - i. Schritt 0: $SSCmac = SSCmac + 1$.
 - ii. Schritt 1: $SSCenc = OS2I(AES_ENC(Kenc, I2OS(SSCmac, 16)))$.
 - iii. Schritt 2: $P = \text{PaddingIso}(M, 16)$.
 - iv. Schritt 3: $C = AES_CBC_ENC(Kenc, SSCenc, P)$.
 - v. Schritt 3: $cipher = '01' || C$.
 2. Option_DES, desSessionkey besitzt, dann gilt mit den Attributen *SSCenc* und *Kenc* aus dem Attribut *SessionkeyContext*
 - i. Schritt 1: $SSCenc = SSCenc + 1$.
 - ii. Schritt 2: $P = \text{PaddingIso}(M, 8)$.

iii. Schritt 3: $C = 3DES_CBC_ENC(K_{enc}, SSC_{enc}, P)$.

iv. Schritt 4: $cipher = '01' \parallel C$.

3. `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: 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 $cipherDO = '86-L_{86}-(02 \parallel C)'$.

v. Setze $macDO = '8E-L_{8E}-T'$.

vi. Setze $cipher = 'A6-L_{A6}-(oidDO \parallel keyDO \parallel cipherDO \parallel macDO)'$.

4. `rsaEncipherOaep` besitzt, dann gilt:

Schritt 1: $C = RSAES_OAEP_ENCRYPT(affectedObject.publicKey, M)$

Schritt 2: $cipher = '85' \parallel C$.

5. `rsaEncipherPKCS1_V1_5` besitzt, dann gilt:

Schritt 1: $C = RSAES_PKCS1_V1_5_ENCRYPT(affectedObject.publicKey, M)$

Schritt 2: $cipher = '81' \parallel C$.

(N091.700) `K_COS`

Wenn `channelContext.keyReferenceList.dataEncipher` leer ist, genau dann MUSS das Kommando auf folgende Art mit einem Schlüssel in den Kommanddaten arbeiten: Das Chiffre `cipher` MUSS wie folgt berechnet werden:

a. Suche in `plainDO` nach einem DO mit Tag = '80' mit der Länge eins. Es gilt $algID_{enc} = \text{Wertfeld dieses DO}$. Wenn $algID_{enc}$ den Wert

b. `rsaEncipherPKCS1_V1_5` besitzt, dann gilt:

1. $plainDO = 'A0-L_{A0}-(algDO \parallel keyDO \parallel mDO)'$.

2. $algDO = '80\ 01\ algID_{enc}'$.

3. $keyDO = '7F49-L_{7F49}-[(81-L_{81}-PuK.n) \parallel (82-L_{82}-PuK.e)]'$.

4. $mDO = '80-L_{80}-M'$.

5. $cipher = '00' \parallel RSAES_PKCS1_V1_5_ENCRYPT(PuK, M)$.

c. `rsaEncipherOaep` besitzt, dann gilt:

1. $plainDO = 'A0-L_{A0}-(algDO \parallel keyDO \parallel mDO)'$.

2. $algDO = '80\ 01\ algID_{enc}'$.

3. $keyDO = '7F49-L_{7F49}-[(81-L_{81}-PuK.n) \parallel (82-L_{82}-PuK.e)]'$.

4. $mDO = '80-L_{80}-M'$.

5. $cipher = '00' \parallel \text{RSAES_OAEP_ENCRYPT}(PuK, M)$.
 - d. $\text{elcSharedSecretCalculation}$ besitzt, dann MUSS gelten (*Hinweis: cipher ist hier identisch zu (N085.068)b.7, (N090.300)c und (N094.400)c definiert*):
 1. $plainDO = 'A0-L_{A0}-(algDO \parallel oidDO \parallel keyDO \parallel mDO)$.
 2. $algDO = '80\ 01\ algID_enc'$.
 3. $oidDO = '06-L_{06}-oid'$.
 4. $keyDO = '7F49-L_{7F49}-(86-L_{86}-PO_B)'$.
 5. $mDO = '80-L_{80}-M'$.
 6. Die oid aus der Kommandonachricht wird gemäß Tabelle 271 in Domainparameter übersetzt, die im Folgenden mit dP bezeichnet werden.
 7. $(PO_A, C, T) = \text{ELC_ENC}(M, PO_B, dP)$
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:
 8. Setze $keyDO = '7F49-L_{7F49}-(86-L_{86}-PO_A)'$.
 9. Setze $cipherDO = '86-L_{86}-(02 \parallel C)'$.
 10. Setze $macDO = '8E-L_{8E}-T'$.
 11. Setze $cipher = 'A6-L_{A6}-(oidDO \parallel keyDO \parallel cipherDO \parallel macDO)$.
- (N091.800) Diese Anforderung ist absichtlich leer. Die in einer früheren Version hier enthaltenen Anforderungen sind nun in (N091.700) enthalten.
- (N091.900) K_COS
Als Datenfeld der Antwortnachricht MUSS $cipher$ verwendet werden.
- (N092.000) K_COS
Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
- (N092.100) K_COS
Für die Priorität der Trailer gilt:
 - a. Die Priorität der Trailer in Tabelle 202 ist herstellerspezifisch.
 - b. Jeder Trailer in Tabelle 202 MUSS eine höhere Priorität als NoError haben.

14.8.5 PSO Hash

- (N092.200) K_COS
Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-8]
- a. unterstützen oder
 - b. ablehnen.

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 14.9.9.11). 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 Chiffprat mittels eines RSA-Schlüssels entschlüsselt und anschließend mit einem anderen RSA-Schlüssels verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos vier Parameter:

(N092.300) K_externeWelt {K_Karte}

Der Parameter C_{in} enthält die umzuschlüsselnden Daten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.

(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).

(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 {rsaEncipherPKCS1_V1_5, rsaEncipherOaep} verwendet werden.

(N092.600) K_externeWelt {K_Karte}

Der Parameter $length$ bestimmt die Länge der erwarteten Antwortdaten. Der Wert von $length$ MUSS gleich WildCardExtended sein.

(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.

(N092.800) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 203 verwendet werden.

Tabelle 203: PSO Transcipher, Umschlüsseln von Daten 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 Chiffprat
P2	‘B8’	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	‘XX...XX’	<i>cipherIN</i> , DER codiertes Datenfeld
Le	‘0000’	<i>length</i> , 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 Chiffprat mittels eines RSA-Schlüssels entschlüsselt und anschließend mit einem ELC-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos fünf Parameter:

(N092.820) K_externeWelt {K_Karte}

Der Parameter C_{in} enthält die umzuschlüsselnden Daten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.

(N092.821) K_externeWelt {K_Karte}

Der Parameter oid_{out} enthält einen Objektidentifizierer, der die zu verwendende elliptische Kurve für die Verschlüsselung referenziert. Der Parameter oid_{out} MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.

(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).

(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.

(N092.826) K_externeWelt {K_Karte}

Der Parameter $length$ bestimmt die Länge der erwarteten Antwortdaten. Der Wert von $length$ MUSS gleich WildCardExtended sein.

(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.

(N092.830) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 204 verwendet werden.

Tabelle 204: PSO Transcipher, Umschlüsseln von Daten 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 Chifftrat
P2	'B8'	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	'XX...XX'	<i>cipherIN</i>
Le	'0000'	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten

14.8.6.3 Use Case Umschlüsseln von Daten mittels ELC

In dieser Variante wird ein Chifftrat mittels eines ELC-Schlüssels entschlüsselt und anschließend mit einem anderen ELC-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos acht Parameter:

(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).

(N092.902) K_externeWelt {K_Karte}

Der Parameter oid_{in} enthält einen Objektidentifizierer, der die zu verwendende elliptische Kurve für die Entschlüsselung referenziert. Der Parameter oid_{in} MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.

(N093.000) K_externeWelt {K_Karte}

Der Parameter C_{in} enthält die umzuschlüsselnden Daten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.

(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.

(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).

(N093.202) K_externeWelt {K_Karte}

Der Parameter oid_{out} enthält einen Objektidentifizierer, der die zu verwendende elliptische Kurve für die Verschlüsselung referenziert. Der Parameter oid_{out} MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.

(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.

(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.

(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.

(N093.600) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 205 verwendet werden.

Tabelle 205: PSO Transcipher, Umschlüsseln von Daten 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 Chiffprat
P2	'B8'	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	'XX...XX'	<i>cipherIN</i>
Le	<i>length</i>	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 Chiffprat mittels eines ELC-Schlüssels entschlüsselt und anschließend mit einem anderen RSA-Schlüssel verschlüsselt.

In dieser Variante enthält die APDU des PSO Transcipher-Kommandos sieben Parameter:

(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).

(N093.621) K_externeWelt {K_Karte}

Der Parameter oid_{in} enthält einen Objektidentifizierer, der die zu verwendende elliptische Kurve für die Entschlüsselung referenziert. Der Parameter oid_{in} MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.

(N093.622) K_externeWelt {K_Karte}

Der Parameter C_{in} enthält die umzuschlüsselnden Daten. Der Parameter C_{in} ist ein Oktettstring mit beliebigem Inhalt.

(N093.624) K_externeWelt {K_Karte}

Der Parameter T_n enthält einen MAC, der die Integrität von C schützt. Der Parameter T_n ist ein Oktettstring, dessen Länge und Inhalt so gewählt werden SOLL, dass bei der MAC-Prüfung kein Fehler auftritt.

(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).

(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 {rsaEncipherPKCS1_V1_5, rsaEncipherOaep} verwendet werden.

(N093.630) K_externeWelt {K_Karte}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.

(N093.632) K_externeWelt {K_Karte}

Die Parameter PO_A , oid_n , C_{in} , T_n , PuK und $algID_{enc}$ MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N094.100), (N094.200) und (N094.400) spezifiziert.

(N093.634) K_externeWelt {K_Karte}

Es MUSS eine Case 4 Kommando-APDU gemäß 11.7.4 über die Schnittstelle „In-

terpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 206 verwendet werden.

Tabelle 206: PSO Transcipher, Umschlüsseln von Daten 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 Chiffprat
P2	'B8'	Beschreibung der Kommandodaten, hier CRT für Verschlüsselung
Data	'XX...XX'	<i>cipherIN</i>
Le	'0000'	length, Anzahl der erwarteten Oktette in den Antwortdaten

14.8.6.5 Antwort der Karte auf Umschlüsseln von Daten

Tabelle 207: PSO Transcipher Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	cipherOUT	Chiffprat
Trailer	Inhalt	Beschreibung
'90 00'	NoError	Erfolgreiche Umschlüsselungsoperation

Tabelle 208: 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 (138): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N093.700) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.6.6 Kommandoabarbeitung innerhalb der Karte

(N093.800) K_COS

- Das COS MUSS die PSO Transcipher-Variante aus 14.8.6.1, 14.8.6.2, 14.8.6.3 und 14.8.6.4 unterstützen.
- Das COS KANN weitere PSO Transcipher-Varianten
 - unterstützen oder
 - ablehnen.

(N093.900) K_COS

Wenn *channelContext.keyReferenceList.dataDecipher*

- a. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
- b. nicht leer ist, dann wird *affectedObject* gleich dem Schlüssel gesetzt, der durch *SearchSecretKey(currentFolder, keyReferenceList.dataDecipher.keyReference, keyReferenceList.dataDecipher.algID)* bezeichnet wird. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls 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.

(N094.000) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.

(N094.010) K_COS

Wenn *affectedObject.keyAvailable* den Wert *False* besitzt, genau dann MUSS das Kommando mit dem Trailer *KeyInvalid* terminieren.

(N094.100) K_COS

Die Kommandodaten werden wie folgt aufgeteilt:

- a. *cipherIN* = *cipher* || *plainDO*.
- b. *cipher* MUSS ein DO mit Tag = 'A6' sein.
- c. *plainDO* MUSS ein DO mit Tag = 'A0' sein.

(N094.200) K_COS

Der Klartext wird wie folgt berechnet:

Wenn *channelContext.keyReferenceList.dataDecipher.algID* den Wert

- a. *rsaDecipherPKCS1_V1_5* besitzt, dann gilt:
 1. *cipher* = 'A6-L_{A6}-(*cipherDO_{in}*)'.
 2. *cipherDO_{in}* = '86-L₈₆-(00 || *C_{in}*)'.
 3. *M* = *RSAES_PKCS1_V1_5_DECRYPT(affectedObject.privateRsaKey, C_{in})*.
Wenn diese Funktion mit dem Fehler „ERROR“ terminiert, genau dann MUSS das Kommando mit dem Trailer *WrongCiphertext* terminieren.
- b. *rsaDecipherOaep* besitzt, dann gilt:
 1. *cipher* = 'A6-L_{A6}-(*cipherDO_{in}*)'.
 2. *cipherDO_{in}* = '86-L₈₆-(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.

- c. `elcSharedSecretCalculation` besitzt, dann gilt (*Hinweis: 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.

(N094.300) K_COS

Suche in *plainDO* nach einem DO mit Tag = '80' dieses DO MUSS die Länge eins haben. Es gilt $algID_enc = \text{Wertfeld des DO mit Tag = '80'}$.

(N094.400) K_COS

Das Chiffre *cipherOUT* MUSS wie folgt aus *M* berechnet werden: Wenn $algID_enc$ den Wert

- a. `rsaEncipherPKCS1_V1_5` 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. $cipherOUT = '00' || RSAES_PKCS1_V1_5_ENCRYPT(PuK, M)$.
Wenn diese Funktion mit dem Fehler "ERROR" terminiert, dann MUSS das Kommando mit dem Trailer `MessageTooLong` terminieren.

- b. `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.

- c. `elcSharedSecretCalculation` besitzt, dann gilt (*Hinweis: 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 Tabelle 271 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 $cipherOUT = 'A6-L_{A6}-(oidDO_{out} || keyDO_{out} || cipherDO_{out} || macDO_{out})'$.

(N094.500) K_COS

Als Datenfeld der Antwortnachricht MUSS *cipherOUT* verwendet werden.

(N094.600) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N094.700) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 208 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 208 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 14.9.9.10). 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 Import RSA-Schlüssels mittels Zertifikat, Option_RSA_CVC

In dieser Variante enthält die APDU des PSO Verify Certificate-Kommandos zwei Parameter:

(N094.800) K_externeWelt {K_Karte}, Option_RSA_CVC

Der Parameter *certificateContent* enthält Attributsinformationen für das Schlüsselobjekt.

(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.

(N094.820) K_externeWelt {K_Karte}, Option_RSA_CVC

Der in *certificateContent* enthaltene Wert für CHA MUSS aus Element der Anwendungsspezifikation genannten Menge sein.

(N094.900) K_externeWelt {K_Karte}, Option_RSA_CVC

Der Parameter *signature* enthält die von einer CA erstellte Signatur über die Attributsinformationen.

(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.

(N095.100) K_externeWelt {K_Karte}, Option_RSA_CVC

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 209 verwendet werden.

Tabelle 209: 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’	<i>certificate</i>

14.8.7.2 Use Case Import ELC-Schlüssels mittels Zertifikat

In dieser Variante enthält die APDU des PSO Verify Certificate-Kommandos zwei Parameter:

(N095.200) K_externeWelt {K_Karte}

Der Parameter *certificateContent* enthält alle Attribute für das Schlüsselobjekt.

(N095.300) K_externeWelt {K_Karte}

Der Parameter *signature* enthält die von einer CA erstellte Signatur über die Attribute.

(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 spezifiziert.

(N095.410) K_COS

Der öffentliche Punkt *P* in 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 Tabelle 210 genannten Schlüsselwörter gelten.

Tabelle 210: 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

(N095.500) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „In-

terpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 211 verwendet werden.

Tabelle 211: 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’	<i>certificate</i>

14.8.7.3 Antwort der Karte auf Vergleich eines Benutzergeheimnisses

Tabelle 212: PSO Verify Certificate Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘63 Cx’	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
‘90 00’	NoError	Erfolgreiche Zertifikatsprüfung

Tabelle 213: 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 Signaturprüfschlüssels ist abgelaufen 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 (139): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N095.600) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.7.4 Kommandoabarbeitung innerhalb der Karte

(N095.700) K_COS

- a. Das COS MUSS die PSO Verify Certificate-Variante aus
 1. Option_RSA_CVC, 14.8.7.1 und
 2. 14.8.7.2 unterstützen.
- b. Das COS KANN weitere PSO Verify Certificate-Varianten

1. unterstützen oder
2. ablehnen.

(N095.800) K_COS

Wenn *channelContext.keyReferenceList.verifyCertificate*

- a. leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
- b. nicht leer ist, dann wird *affectedObject* = SearchPublicKey(
 currentFolder,
 keyReferenceList.verifyCertificate.keyReference,
 verifyCertificate
) gesetzt. Gemäß 9.2.3.2 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Genau dann MUSS das Kommando mit dem Trailer KeyNotFound terminieren.

(N095.810) K_COS

Falls *affectedObject* ein Attribut *expirationDate* besitzt und *affectedObject.CHAT.flagList* nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. b0 b1 ist ungleich 11₂) und *affectedObject.expirationDate* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.

(N095.820) K_COS

Wenn AccessRuleEvaluation(*affectedObject*, *CLA*, *INS*, *P1*, *P2*) den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatus-NotSatisfied terminieren.

(N095.900) K_COS

Extraktion von Schlüsselinformationen:

- a. Option_RSA_CVC, Wenn *affectedObject.publicKey* vom Typ *publicRsaKey* ist, dann gilt:
 1. *certificate* = '5F37–L_{5F37}–signature || 5F38–L_{5F38}–certificateContent'.
 2. (*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.
 3. Die Schlüsselattribute werden gemäß 7.1.2 aus der Nachricht *M* extrahiert und ein öffentliches Schlüsselobjekt *pukObj* gebildet.
 4. 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:
 - i. *pukObj.oid* = OID.
 - ii. *pukObj.publicKey.n* = Modulus.
 - iii. *pukObj.publicKey.e* = öffentlicherExponent.
 - iv. *pukObj.keyIdentifier* = CHR.
 - v. *pukObj.lifeCycleStatus* = „Operational state (active)“

- vi. *pukObj.accessRules* = *affectedObject.accessRulesPublicSignatureVerificationObject*.
 - vii. *pukObj.accessRulesPublicSignatureVerificationObject* = *affectedObject.accessRulesPublicSignatureVerificationObject*.
 - viii. *pukObj.accessRulesPublicAuthenticationObject* = *affectedObject.accessRulesPublicAuthenticationObject*.
5. 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:
- i. *pukObj.CHA* = CHA.
 - ii. *pukObj.oid* = OID.
 - iii. *pukObj.publicKey.n* = Modulus.
 - iv. *pukObj.publicKey.e* = öffentlicherExponent.
 - v. *pukObj.keyIdentifier* = CHR.
 - vi. *pukObj.lifeCycleStatus* = „Operational state (active)“
 - vii. *pukObj.accessRules* = *affectedObject.accessRulesPublicAuthenticationObject*.
- b. Wenn *affectedObject.publicKey* vom Typ *publicElcKey* ist, dann gilt:
1. CPI:
- i. Das COS MUSS den Wert CPI = '70' = 112 unterstützen.
 - ii. Das COS KANN weitere Werte für CPI
 - A. unterstützen oder
 - B. ablehnen.
2. *certificate* = '7F4E-L_{7F4E}- *certificateContent* || 5F37-L_{5F37}- *signature*'.
3. 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*').
4. *signature* wird wie folgt in *R* und *S* aufgeteilt:
signature = *R* || *S*, mit OctetLength(*R*) = OctetLength(*S*).
5. *out* = ELC_VER_SIG(
 affectedObject.publicElcKey,
 R,
 S,
 hash
)
- Falls diese Operation mit einem Fehler abbricht, oder *out* gleich False ist, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
6. Falls die OID im CHAT aus *certificateContent* verschieden ist von der aus *affectedObject.publickey.accessRights*, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.

7. Das COS MUSS das Wertfeld von CED als vorzeichenlose ganze Zahl interpretieren, gemäß $ced = OS2I(\text{Wertfeld_von_CED})$.
8. Das COS MUSS das Wertfeld von CXD als vorzeichenlose ganze Zahl interpretieren, gemäß $cx d = OS2I(\text{Wertfeld_von_CXD})$.
9. Falls CED aus *certificateContent* größer ist als CXD aus *certificateContent*, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
10. Das COS MUSS den Wert *effectiveFlagList* wie folgt bilden:
$$\text{effectiveFlagList} = \text{affectedObject.CHAT.flagList AND} \\ (\text{flagList aus certificateContent})$$
11. Falls *effectiveFlagList*
 - i. die Rolle Root-CA-Schlüssel anzeigt (d.h. $b_0 b_1$ ist gleich 11_2), dann DARF ein Import NICHT daran scheitern, CXD aus *certificateContent* größer ist als *affectedObject.expirationDate*.
 - ii. nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. $b_0 b_1$ ist ungleich 11_2) und CXD aus *certificateContent* größer ist als *affectedObject.expirationDate*, dann MUSS das Kommando mit dem Trailer VerificationError terminieren.
12. Falls *effectiveFlagList*
 - i. die Rolle Root-CA-Schlüssel anzeigt (d.h. $b_0 b_1$ ist gleich 11_2), dann DARF ein Import NICHT daran scheitern, dass CXD aus *certificateContent* kleiner als *pointInTime* ist (siehe (N019.900)j).
 - ii. nicht die Rolle Root-CA-Schlüssel anzeigt (d.h. $b_0 b_1$ ist ungleich 11_2) und CXD aus *certificateContent* kleiner als *pointInTime* ist (siehe (N019.900)j), dann MUSS das Kommando mit dem Trailer KeyExpired terminieren.
13. Falls CED aus *certificateContent* größer als *pointInTime* ist (siehe (N019.900)j), dann MUSS *pointInTime* mit Transaktionsschutz auf den Wert CED gesetzt werden.
14. Die Schlüsselattribute werden gemäß [gemSpec_PKI] aus *certificateContent* extrahiert und ein öffentliches Schlüsselobjekt *pukObj* gebildet.
15. Falls das höchstwertige Bit von *effectiveFlagList*
 - i. gesetzt ist, dann MUSS *pukObj* ein öffentliches Signaturprüfobjekt sein (siehe 8.6.4.1).
 - ii. nicht gesetzt ist, dann MUSS *pukObj* ein öffentliches Authentisierungsobjekt sein (siehe 8.6.4.2).
16. Die Attribute von *pukObj* sind mit den Definitionen aus [gemSpec_PKI] wie folgt zu setzen:
 - i. Domainparameter: Falls das Wertfeld von DO'86' eine Länge von
 - A. '41' = 65 hat, gilt $PuK.domainParameter = \text{brainpoolP256r1}$.
 - B. '61' = 97 hat, gilt $PuK.domainParameter = \text{brainpoolP384r1}$.
 - C. '81' = 129 hat, gilt $PuK.domainParameter = \text{brainpoolP512r1}$.
 - ii. $PuK.P = OS2P(\text{Wertfeld von DO'86'})$

PuK.domainParameter

).

- iii. *pukObj.keyIdentifier* = CHR gemäß [gemSpec_PKI#6.7.2.4].
 - iv. *pukObj.lifeCycleStatus* = „Operational state (active)“
 - v. *pukObj.publicElcKey* = *PuK*.
 - vi. *pukObj.oid* = *OID_{PuK}*.
Hinweis: Anhand von pukObj.oid ist erkennbar, ob pukObj ein Signaturprüfobjekt oder ein Authentisierungsobjekt ist.
 - vii. Falls *pukObj* ein öffentliches Signaturprüfobjekt ist, dann gilt:
 - A. *pukObj.accessRules* = *affectedObject.accessRulesPublicSignatureVerificationObject*.
 - B. *pukObj.accessRulesPublicSignatureVerificationObject* = *affectedObject.accessRulesPublicSignatureVerificationObject*.
 - C. *pukObj.accessRulesPublicAuthenticationObject* = *affectedObject.accessRulesPublicAuthenticationObject*.
 - viii. Falls *pukObj* ein öffentliches Authentisierungsobjekt ist, dann gilt:
pukObj.accessRules = *affectedObject.accessRulesPublicAuthenticationObject*.
 - ix. *pukObj.CHAT.OID_{flags}* = *affectedObject.CHAT.OID_{flags}*.
 - x. *pukObj.CHAT.flagList* = *effectiveFlagList*
 - xi. *pukObj.expirationDate* = Wertfeld des Datenobjektes CXD.
- c. Falls *affectedObject.keyIdentifier* ungleich CAR aus dem importierten Zertifikat ist, genau dann MUSS das Kommando mit dem Trailer *InconsistentKeyReference* terminieren.
- d. Das Objekt *pukObj* wird wie folgt gespeichert:
- 1. *persistent* = StoreInCache(„Ordner dem *affectedObject* zugeordnet ist“, *pukObj*)

(N096.000) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer *UpdateRetryWarning* wählen.

(N096.100) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- a. entweder als Trailer *MemoryFailure* verwendet werden,
- b. oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N096.200) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer *NoError* gewählt werden.

(N096.300) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 213 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 213 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, 15.4.4) ausgehandelt. Checksumme und geschützte Daten sind als Parameter in der Kommandonachricht enthalten.

14.8.8.1 Use Case Prüfung 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:

(N096.340) K_externeWelt {K_Karte}

Der Parameter *data* enthält die geschützten Daten. Der Parameter *data* ist ein Oktettstring mit beliebigem Inhalt.

(N096.342) K_externeWelt {K_Karte}

Der Parameter *mac* enthält eine kryptographische Checksumme. Der Parameter *mac* ist ein Oktettstring mit beliebigem Inhalt, dessen Länge acht Oktette betragen MUSS.

Hinweis (140): 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.

(N096.344) K_externeWelt {K_Karte}

Die Parameter *data* und *mac* MÜSSEN im Datenfeld der Kommandonachricht enthalten sein. Die Codierung wird in (N096.366)a spezifiziert.

(N096.346) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 214 verwendet werden.

Tabelle 214: 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'	<i>inputTemplate</i> , DER codiertes Datenfeld

14.8.8.2 Antwort der Karte auf Berechnen einer kryptographischen Checksumme

Tabelle 215: PSO Verify Cryptographic Checksum Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'90 00'	NoError	erfolgreiche Verifizierung eines MAC

Tabelle 216: 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 angegebenen Algorithmus nicht
'6A 88'	KeyNotFound	kein Schlüssel für MAC-Berechnung vorhanden

Hinweis (141): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N096.350) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.8.3 Kommandoabarbeitung innerhalb der Karte

(N096.360) K_COS, Option_Kryptobox, Option_PACE_PCD

- Das COS MUSS die PSO Verify Cryptographic Checksum-Variante aus 14.8.8.1 unterstützen.
- Das COS KANN weitere PSO Verify Cryptographic Checksum-Varianten
 - unterstützen oder
 - ablehnen.

(N096.362) K_COS

Wenn *channelContext.keyReferenceList.macCalculation*

- leer ist, genau dann MUSS das Kommando mit dem Trailer NoKeyReference terminieren.
- nicht leer ist, dann wird *affectedObject = SearchKey(currentFolder, keyReferenceList.macCalculation.keyReference, keyReferenceList.macCalculation.algID)* gesetzt. Gemäß 9.2.3 und (N104.300) ist es möglich, dass die Schlüsselsuche nicht erfolgreich ist. Falls die Schlüsselsuche den Fehler
 - KeyNotFound terminieren.
 - notSupported meldet, genau dann MUSS das Kommando mit dem Trailer UnsupportedFunction terminieren.

(N096.364) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert False zurückliefert, genau dann MUSS das Kommando mit dem Trailer SecurityStatusNotSatisfied terminieren.

(N096.366) K_COS

Es gilt:

- inputTemplate = '80-L₈₀-data || 8E-L_{8E}-mac'*

(N096.368) K_COS

Mit den Attributen *SSCmac* und *Kmac* aus dem Attribut *SessionkeyContext* gilt:
Falls *keyReferenceList.macCalculation.algID* den Wert

- a. aesSessionkey besitzt, gilt:
 - 1. $SSCmac = SSCmac + 1$.
 - 2. $result = \text{VerifyCMAC_IsoPadding}(Kmac, mac, \text{I2OS}(SSCmac, 16) || data)$.
- b. Option_DES, desSessionkey besitzt, gilt:
 - 1. $SSCmac = SSCmac + 1$.
 - 2. $result = \text{VERIFY_Retail_MAC}(Kmac, mac, \text{I2OS}(SSCmac, 8) || data)$.

(N096.370) K_COS

Falls *result* den Wert

- a. INVALID besitzt, dann
 - 1. MUSS als Trailer VerificationError verwendet werden und
 - 2. die Sessionkeys MÜSSEN mittels clearSessionkeys() gelöscht werden.
- b. VALID besitzt, genau dann MUSS als Trailer NoError verwendet werden.

(N096.372) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 216 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 216 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:

(N096.380) K_externeWelt {K_Karte}

Der Parameter *oid* enthält einen Objektidentifizierer, der die zu verwendende elliptische Kurve referenziert. Der Parameter *oid* MUSS aus der in Tabelle 271 genannten Menge gewählt werden und eine elliptische Kurve aus der Menge in (N002.500) referenzieren.

(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 (N004.500)a).

(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` \Rightarrow `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.

(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.

(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.

(N096.388) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4 Kommando-APDU MÜSSEN die Angaben aus Tabelle 217 verwendet werden.

Tabelle 217: 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'	<i>signatureTemplate</i> , DER codiertes Datenfeld

14.8.9.2 Antwort der Karte auf Prüfen einer digitalen Signatur

Tabelle 218: PSO Verify Digital Signature Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'90 00'	NoError	erfolgreiche Verifizierung einer Signatur

Tabelle 219: PSO Verify Digital Signature Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'6A 80'	VerificationError	Signaturprüfung fehlgeschlagen

Hinweis (142): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N096.390) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.8.9.3 Kommandoabarbeitung innerhalb der Karte

(N096.392) K_COS

- a. Das COS MUSS die PSO Verify Digital Signature-Variante aus 14.8.9.1 unterstützen.
- b. Das COS KANN weitere PSO Verify Digital Signature-Varianten
 1. unterstützen oder
 2. ablehnen.

(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 Tabelle 271 in Domainparameter übersetzt, die im Folgenden mit dP bezeichnet werden.

(N096.396) K_COS

Falls $ELC_VER_SIG(OS2P(PO_B, dP), R, S, hash)$ den Wert

- a. False besitzt, genau dann MUSS als Trailer VerificationError verwendet werden.
- b. True besitzt, genau dann MUSS als Trailer NoError verwendet werden.

(N096.398) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 219 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 219 MUSS eine höhere Priorität als NoError haben.

14.9 Verschiedenes

14.9.1 ENVELOPE

(N096.400) K_COS

Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]

- a. unterstützen oder
- b. ablehnen.

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 (143): 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 Fingerprint über das COS berechnen.

In dieser Variante enthält die APDU des FINGERPRINT-Kommandos zwei Parameter:

(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* beträgt 128 Oktett.

(N096.452) K_externeWelt {K_Karte}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* beträgt WildCardShort.

(N096.454) K_externeWelt {K_Karte}

Es MUSS eine Case 4S Kommando-APDU gemäß 11.7.4.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 4S Kommando-APDU MÜSSEN die Angaben aus Tabelle 220 verwendet werden.

Tabelle 220: 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’	<i>prefix</i>
Le	‘00’	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten

14.9.2.2 Antwort der Karte auf Fingerprintberechnung

Tabelle 221: FINGERPRINT Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
‘xx...xx’	<i>fingerprint</i>	Fingerprint des COS
‘90 00’	NoError	Erfolgreiche Fingerprintberechnung

Tabelle 222: FINGERPRINT Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
‘69 82’	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis (144): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N096.469) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.9.2.3 Kommandoabarbeitung innerhalb der Karte

(N096.470) K_COS

- a. Das COS MUSS die FINGERPRINT-Variante aus 14.9.2.1 unterstützen.
- b. Das COS KANN weitere FINGERPRINT-Varianten
 1. unterstützen oder
 2. ablehnen.

(N096.472) K_COS

Als *affectedObject* MUSS *currentFolder* verwendet werden.

(N096.474) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.

(N096.476) K_COS

Repräsentant *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.

(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.

(N096.480) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer *NoError* gewählt werden.

(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. Das betroffene Schlüsselobjekt wird zuvor ausgewählt. Dies geschieht durch ein *MANAGE SECURITY ENVIRONMENT*-Kommando (siehe 14.9.9.9). Zusätzlich ist es möglich, den zu generierenden Schlüssel im Kommando zu referenzieren.

14.9.3.1 Use Case Generierung, ohne Überschreiben, ohne Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MANAGE SECURITY ENVIRONMENT-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:

(N096.500) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '84' gewählt werden.

(N096.600) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 223 verwendet werden.

Tabelle 223: 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'	<i>operationMode</i> = Schlüsselgenerierung, falls kein Schlüssel vorhanden
P2	'00'	betroffenes Objekt via <i>channelContext.keyReferenceList</i>

14.9.3.2 Use Case Generierung, 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:

(N096.640) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '84' gewählt werden.

(N096.642) K_externeWelt {K_Karte}

Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.

(N096.644) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 224 verwendet werden.

Tabelle 224: 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'	<i>operationMode</i> = Schlüsselgenerierung, falls kein Schlüssel vorhanden
P2	'XX'	<i>keyReference</i> bestimmt betroffenes Schlüsselobjekt

14.9.3.3 Use Case Generierung, ggf. Überschreiben, ohne Referenz, ohne Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MANAGE SECURITY ENVIRONMENT-Kommando ausgewählt und mit Schlüsseldaten befüllt. Falls bereits Schlüsseldaten vorhanden sind, dann werden diese überschrieben. Die APDU des GAKP-Kommandos enthält hier einen Parameter:

(N096.650) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C4' gewählt werden.

(N096.652) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 225 verwendet werden.

Tabelle 225: 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'	<i>operationMode</i> = Schlüsselgenerierung, ggf. Überschreiben
P2	'00'	betroffenes Objekt via <i>channelContext.keyReferenceList</i>

14.9.3.4 Use Case Generierung, 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:

(N096.660) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C4' gewählt werden.

(N096.662) K_externeWelt {K_Karte}

Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.

(N096.664) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 226 verwendet werden.

Tabelle 226: 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'	<i>operationMode</i> = Schlüsselgenerierung, ggf. Überschreiben
P2	'XX'	<i>keyReference</i> bestimmt betroffenes Schlüsselobjekt

14.9.3.5 Use Case Auslesen vorhandener Schlüssel, ohne Referenz

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MANAGE SECURITY ENVIRONMENT-Kommando ausgewählt. Die dort vorhandenen Schlüsseldaten werden ausgelesen. Die APDU des GAKP-Kommandos enthält hier zwei Parameter:

(N096.700) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.

(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* in der Antwortnachricht enthalten ist.

(N096.900) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 227 verwendet werden.

Tabelle 227: 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'	<i>operationMode</i> = Auslesen eines öffentlichen Schlüssels
P2	'00'	betroffenes Objekt via <i>channelContext.keyReferenceList</i>
Le	<i>length</i>	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:

(N096.940) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.

(N096.942) K_externeWelt {K_Karte}

Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.

(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* in der Antwortnachricht enthalten ist.

(N096.946) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 228 verwendet werden.

Tabelle 228: 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'	<i>operationMode</i> = Auslesen eines öffentlichen Schlüssels
P2	'XX'	<i>keyReference</i> bestimmt betroffenes Schlüsselobjekt
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.9.3.7 Use Case Generierung, ohne Überschreiben, ohne Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MANAGE SECURITY ENVIRONMENT-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:

(N097.000) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '80' gewählt werden.

(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* in der Antwortnachricht enthalten ist.

(N097.200) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 229 verwendet werden.

Tabelle 229: 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'	<i>operationMode</i> = Schlüsselgenerierung falls nicht vorhanden, Ausgabe
P2	'00'	betroffenes Objekt via <i>channelContext.keyReferenceList</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.9.3.8 Use Case Generierung, 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 Schlüsselpaars 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:

(N097.240) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '80' gewählt werden.

(N097.242) K_externeWelt {K_Karte}

Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.

(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* in der Antwortnachricht enthalten ist.

(N097.246) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 230 verwendet werden.

Tabelle 230: 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'	<i>operationMode</i> = Schlüsselgenerierung falls nicht vorhanden, Ausgabe
P2	'XX'	<i>keyReference</i> bestimmt betroffenes Schlüsselobjekt
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.9.3.9 Use Case Generierung, ggf. Überschreiben, ohne Referenz, mit Ausgabe

In dieser Variante wird das betroffene Schlüsselobjekt durch ein MANAGE SECURITY ENVIRONMENT-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:

(N097.250) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C0' gewählt werden.

(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* in der Antwortnachricht enthalten ist.

(N097.254) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 231 verwendet werden.

Tabelle 231: 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'	<i>operationMode</i> = Schlüsselgenerierung, ggf. Überschreiben, Ausgabe
P2	'00'	betroffenes Objekt via <i>channelContext.keyReferenceList</i>
Le	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.9.3.10 Use Case Generierung, 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:

(N097.260) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'C0' gewählt werden.

(N097.262) K_externeWelt {K_Karte}

Der Parameter *keyReference* referenziert das von der Aktion betroffene Schlüsselobjekt und MUSS gemäß (N099.600) gewählt werden.

(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* in der Antwortnachricht enthalten ist.

(N097.266) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 232 verwendet werden.

Tabelle 232: 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'	<i>operationMode</i> = Schlüsselgenerierung, ggf. Überschreiben, Ausgabe
P2	'XX'	<i>keyReference</i> bestimmt betroffenes Schlüsselobjekt
Le	<i>length</i>	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. Deshalb sind solche nicht zwingend zu unterstützen.

Tabelle 233: 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, Ausgabe Schlüsselgenerierung, falls kein Schlüssel vorhanden, keine Ausgabe Schlüsselgenerierung, ggf. Überschreiben, Ausgabe Schlüsselgenerierung, ggf. Überschreiben, keine Ausgabe
P2	'00' sonst	betroffenes Objekt via <i>channelContext.keyReferenceList</i> <i>keyReference</i> bestimmt betroffenes Schlüsselobjekt
Le	<i>length</i>	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 Schlüsselgenerierung

Tabelle 234: GAKP Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>publicKeyDO</i>	Abwesend, oder öffentlicher Schlüssel
Trailer	Inhalt	Beschreibung
'63 Cx'	UpdateRetryWarning	Wie NoError, aber Schreibschwierigkeiten
'90 00'	NoError	Erfolgreiche Operation

Tabelle 235: GAKP 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 (145): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N097.300) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.9.3.13 Kommandoabarbeitung innerhalb der Karte

(N097.400) K_COS

a. Das COS MUSS

1. für private ELC-Schlüssel (siehe 8.2.3.2) die GAKP-Varianten aus 14.9.3.1, 14.9.3.2, 14.9.3.3, 14.9.3.4, 14.9.3.5, 14.9.3.6, 14.9.3.7, 14.9.3.8 14.9.3.9 und 14.9.3.10 unterstützen.
2. für private RSA-Schlüssel (siehe 8.2.3.1) die GAKP-Varianten aus 14.9.3.5 und 14.9.3.6 unterstützen.
3. Option_RSA_KeyGeneration:
die GAKP-Varianten aus 14.9.3.1, 14.9.3.2, 14.9.3.3, 14.9.3.4, 14.9.3.5, 14.9.3.6, 14.9.3.7, 14.9.3.8 14.9.3.9 und 14.9.3.10 unterstützen.

b. Das COS KANN weitere GAKP-Varianten

1. unterstützen oder
2. ablehnen.

(N097.500) K_COS

Das vom Kommando betroffene Schlüsselobjekt wird wie folgt bestimmt:

a. Falls der Parameter P2 gleich '00' ist, dann gilt:

Falls das Attribut *channelContext.keyReferenceList.signatureCreation*

1. leer ist, dann MUSS das COS die Bearbeitung dieses Kommandos mit einem beliebigen Trailer beantworten. In diesem Fall ist es zulässig, dass das COS mit einem beliebigen anderen privaten Schlüsselobjekt arbeitet.

2. nicht leer ist, gilt:

keyReference = keyReferenceList.signatureCreation.keyReference

b. Falls der Parameter P2 ungleich '00' ist, MUSS *keyReference = P2* gelten.

c. Es wird *affectedObject = SearchSecretKey(*

currentFolder,
keyReference,
Wildcard

) gesetzt. Gemäß 9.2.3 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. Der Fehler *notSupported* ist wegen der Wildcard-Suche nicht möglich.

(N097.600) K_COS

Wenn *AccessRuleEvaluation(affectedObject, CLA, INS, P1, P2)* den Wert *False* zurückliefert, genau dann MUSS das Kommando mit dem Trailer *SecurityStatus-NotSatisfied* terminieren.

(N097.650) K_COS

Wenn *operationMode* Element der Menge { '80', '84' } ist und das Attribut *keyAvailable* den Wert *True* hat, genau dann MUSS das Kommando mit dem Trailer *KeyAlreadyPresent* terminieren.

(N097.700) K_COS

Wenn *operationMode* Element der Menge { '80', '84', 'C0', 'C4' } ist, dann MUSS

- a. ein Schlüsselpaar (*PrK*, *PuK*) erzeugt werden, dessen Eigenschaften zu den Attributen von *affectedObject* und zu (N002.100) passen.
- b. Anschließend MUSS das Attribut *keyAvailable* auf den Wert *True* geändert werden. Die Änderung von *keyAvailable* MUSS mit Transaktionsschutz ausgeführt werden.

(N097.800) K_COS

Wenn *operationMode* Element der Menge {'81'} ist und das Attribut *keyAvailable* den Wert False hat, genau dann MUSS das Kommando mit dem Trailer KeyInvalid terminieren.

(N097.900) K_COS

Wenn das COS feststellt, dass ein Schreibvorgang nicht beim ersten Versuch erfolgreich verlief, genau dann KANN das COS als Trailer UpdateRetryWarning wählen.

(N098.000) K_COS

Wenn ein Schreibvorgang nicht erfolgreich verlief, genau dann MUSS

- entweder als Trailer MemoryFailure verwendet werden,
- oder die Kommandobearbeitung gemäß (N031.940) stoppen.

(N098.100) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N098.200) K_COS

Das DER-TLV codierte Datenobjekt *publicKeyDO* MUSS wie folgt berechnet werden: Falls *PuK* ein

- RSA-Schlüssel ist, dann gilt:

- Setze L_{82} größer gleich $\text{OctetLength}(PuK.e)$, aber kleiner als 128.
- Setze $n = \text{I2OS}(PuK.n, \text{OctetLength}(PuK.n))$.
- Setze $e = \text{I2OS}(PuK.e, L_{82})$.
- $publicKeyDO = '7F49-L_{7F49}-(81-L_{81}-n || 82-L_{82}-e)'$.

- ELC-Schlüssel ist mit dem öffentlichen Punkt P und den Domainparametern dP , dann gilt
 $publicKeyDO = '7F49-L_{7F49}-(86-L_{86}-P2OS(P, dP.L))'$.

(N098.300) K_COS

Für das Datenfeld der Antwortnachricht MUSS gelten:

- Wenn *operationMode* einen Wert aus der Menge {'80', '81', 'C0'} hat, dann enthält das Datenfeld der Antwortnachricht *publicKeyDO*.
- Andernfalls fehlt das Datenfeld der Antwortnachricht.

(N098.400) K_COS

Für die Priorität der Trailer gilt:

- Die Priorität der Trailer in Tabelle 235 ist herstellerspezifisch.
- Jeder Trailer in Tabelle 235 MUSS eine höhere Priorität als UpdateRetryWarning haben.
- 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 Zufallszahl für DES oder RSA Authentisierung

In dieser Variante enthält die APDU des GET CHALLENGE-Kommandos einen Parameter:

(N098.500) K_externeWelt {K_Karte}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich 8 sein.

(N098.600) K_externeWelt {K_Karte}, Option_DES, Option_RSA_CVC

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 236 verwendet werden.

Tabelle 236: 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'	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten, hier acht

14.9.4.2 Use Case Zufallszahl für AES oder ELC Authentisierung

In dieser Variante enthält die APDU des GET CHALLENGE-Kommandos einen Parameter:

(N098.620) K_externeWelt {K_Karte}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich 16 sein.

(N098.625) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 237 verwendet werden.

Tabelle 237: 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'	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten, hier 16

14.9.4.3 Antwort der Karte auf Erzeugen einer Zufallszahl

Tabelle 238: GET CHALLENGE Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>rspData</i>	Zufallszahl
Trailer	Inhalt	Beschreibung
'90 00'	NoError	Erfolgreiche Erzeugung einer Zufallszahl

Tabelle 239: GET CHALLENGE Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
–	–	Derzeit keine Fehlerfälle spezifiziert

Hinweis (146): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N098.700) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.9.4.4 Kommandoabarbeitung innerhalb der Karte

(N098.800) K_COS

- a. Das COS MUSS die GET CHALLENGE-Variante aus 14.9.4.1 und 14.9.4.2 unterstützen.
- b. Das COS KANN weitere GET CHALLENGE-Varianten
 1. unterstützen oder
 2. ablehnen.

(N098.900) K_COS

Zugriffsregeln für das GET CHALLENGE-Kommando:

- a. Das COS KANN die Auswertung von Zugriffsregeln für das GET CHALLENGE-Kommando unterstützen.
- b. Das COS KANN als Zugriffsbedingung für das GET CHALLENGE-Kommando stets ALWAYS verwenden.

(N099.000) K_COS

Es MUSS $rspData = RAND(Ne)$ gesetzt werden.

(N099.100) K_COS

Als Trailer MUSS NoError gewählt werden.

(N099.200) K_COS

Das Datenfeld der Antwortnachricht MUSS $rspData$ sein.

(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 gewisse Sicherheitsanforderungen (siehe (N099.356)b).

14.9.5.1 Use Case Erzeugen kryptographisch sicherer Zufallszahl

In dieser Variante enthält die APDU des GET RANDOM-Kommandos einen Parameter:

(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.

(N099.322) K_externeWelt {K_Karte}

Es MUSS eine Case 2S Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2S Kommando-APDU MÜSSEN die Angaben aus Tabelle 240 verwendet werden.

Tabelle 240: 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	<i>length</i>	Anzahl der erwarteten Oktette in den Antwortdaten

14.9.5.2 Antwort der Karte Erzeugen kryptographisch sichere Zufallszahl

Tabelle 241: GET RANDOM Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
‘xx...xx’	<i>random</i>	Zufallszahl
Trailer	Inhalt	Beschreibung
‘90 00’	NoError	erfolgreiche Erzeugung einer Zufallszahl

Tabelle 242: GET RANDOM Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
‘69 82’	SecurityStatusNotSatisfied	Zugriffsregel nicht erfüllt

Hinweis (147): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N099.340) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.9.5.3 Kommandoabarbeitung innerhalb der Karte

(N099.344) K_COS, Option_logische_Kanäle

- Das COS MUSS die GET RANDOM-Variante aus 14.9.5.1 unterstützen.
- Das COS KANN weitere GET RANDOM-Varianten
 - unterstützen oder
 - ablehnen.

(N099.348) K_COS

Als *affectedObject* MUSS *currentFolder* verwendet werden.

(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.

(N099.356) K_COS

- a. Es MUSS *random* = RAND(Ne) gesetzt werden.
- b. Die Güte der Zufallszahl in *random* MUSS mindestens [BSI-TR-03116-1#3.5] PTG.3 oder K4-DRNG oder DRG.3 entsprechen.

(N099.360) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N099.364) K_COS

Das Datenfeld der Antwortnachricht MUSS *random* sein.

14.9.6 GET RESPONSE

(N099.400) K_COS

Das COS KANN dieses Kommando gemäß [ISO/IEC 7816-4]

- a. unterstützen oder
- b. ablehnen.

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.

(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.
- b. Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS gleich WildCardExtended sein.

(N099.452) K_externeWelt {K_Karte}

Es MUSS eine Case 2E-Kommando-APDU gemäß 11.7.2.2 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2E-Kommando-APDU MÜSSEN die Angaben aus Tabelle 243 verwendet werden.

Tabelle 243: 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'	gemäß [ISO/IEC 7816-4] proprietärer Wert, hier: Alle Arten von öffentlichen Schlüsselobjekten
P2		
Le	'0000'	<i>length</i> , Anzahl der erwarteten Oktette in den Antwortdaten

14.9.7.2 Antwort der Karte auf Auslesen einer Schlüsselliste

Tabelle 244: LIST PUBLIC KEY Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
'xx...xx'	<i>keyReferenceList</i>	Liste mit Referenzen öffentlicher Schlüsselobjekte
Trailer	Inhalt	Beschreibung
'62 00'	DataTruncated	Antwortdaten unvollständig
'90 00'	NoError	Erfolgreiche Leseoperation

Tabelle 245: LIST PUBLIC KEY Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
-	-	derzeit ist kein Fehlerfall spezifiziert

Hinweis (148): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N099.458) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.9.7.3 Kommandoabarbeitung innerhalb der Karte

(N099.460) K_COS

- Das COS MUSS die LIST PUBLIC KEY-Variante aus 14.9.7.1 unterstützen.
- Das COS KANN weitere LIST PUBLIC KEY-Varianten
 - unterstützen oder
 - ablehnen.

(N099.461) K_COS

Zugriffsregeln für das LIST PUBLIC KEY Kommando:

- Das COS KANN die Auswertung von Zugriffsregeln für das LIST PUBLIC KEY Kommando unterstützen.
- Das COS KANN als Zugriffsbedingung für das LIST PUBLIC KEY Kommando stets ALWAYS verwenden.

(N099.462) K_COS

Das Datenfeld *keyReferenceList* der Antwortnachricht MUSS wie folgt erzeugt werden:

- Falls keine Referenz auf ein öffentliches Schlüsselobjekt zurückgemeldet wird, dann MUSS *keyReferenceList* = '' leer sein.
- Für jedes öffentliche Schlüsselobjekt in *persistentPublicKeyList* MUSS ein Eintrag in *keyReferenceList* erfolgen.
- Für ein öffentliches Schlüsselobjekt in *volatileCache* KANN ein Eintrag in *keyReferenceList*
 - erfolgen oder

2. unterbleiben.
- d. Jeder Eintrag in *keyReferenceList* MUSS ein DER-TLV-Datenobjekt *pukReference* sein.
- e. Alle *pukReference* Datenobjekte MÜSSEN konkateniert werden.
- f. 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. Falls 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. Falls 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.

(N099.463) K_COS

Das COS MUSS für *keyReferenceList* eine Mindestlänge von *limitRspSecureMessaging* unterstützen. Falls *keyReferenceList* länger ist als vom COS unterstützt, dann MUSS als Trailer DataTruncated verwendet werden.

- 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.

(N099.464) K_COS

Für den Trailer der Antwort-APDU gilt:

- a. Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.
- b. Für die Priorität der Trailer gilt: DataTruncated MUSS eine höhere Priorität als NoError haben.

(N099.466) Funktionale Zulassung

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* hinzugefügbar 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 beigelegt 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:

(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 zwei Parameter:

(N099.504) K_externeWelt {K_Karte}

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.

(N099.506) K_externeWelt {K_Karte}

Der Parameter *length* bestimmt die Länge der erwarteten Antwortdaten. Der Wert von *length* MUSS 1 = '01' sein.

(N099.508) K_externeWelt {K_Karte}

Es MUSS eine Case 2 Kommando-APDU gemäß 11.7.2.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 2 Kommando-APDU MÜSSEN die Angaben aus Tabelle 246 verwendet werden.

Tabelle 246: 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'	<i>intendedAction</i> , hier Öffnen eines Kanals, Kanalnummer wird vom COS bestimmt.
P2	'00'	
Le	'01'	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:

(N099.510) K_externeWelt {K_Karte}

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.

(N099.512) K_externeWelt {K_Karte}

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.

(N099.514) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interface I/O“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 247 verwendet werden.

Tabelle 247: 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'	<i>intendedAction</i> , 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:

(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.

(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.

(N099.524) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interface I/O“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 248 verwendet werden.

Tabelle 248: 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'	<i>intendedAction</i> , hier Zurücksetzen eines Kanals,

	Inhalt	Beschreibung
P2	'00'	betroffener Kanal wird im CLA-Byte angezeigt.

14.9.8.4 Use Case logischer Reset der Applikationsebene

In dieser Variante enthält die APDU des MANAGE CHANNEL-Kommandos einen Parameter:

(N099.530) K_externeWelt {K_Karte}

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.

(N099.532) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interface I/O“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 249 verwendet werden.

Tabelle 249: 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'	<i>intendedAction</i> , 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 250: MANAGE CHANNEL Antwort-APDU im Erfolgsfall

Daten	Inhalt	Beschreibung
Ka-nalnr.	'XX'	Nummer des soeben geöffneten Kanals
Trailer	Inhalt	Beschreibung
'90 00'	NoError	erfolgreiche Operation

Tabelle 251: MANAGE CHANNEL Antwort-APDU im Fehlerfall

Trailer	Inhalt	Beschreibung
'69 81'	NoMoreChannelsAvailable	kein weiterer logischen Kanäle verfügbar

Hinweis (149): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes in Abbildung 1 entdeckt wurden.

(N099.540) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.9.8.6 Kommandoabarbeitung innerhalb der Karte

(N099.542) Kommandounterstützung

a. K_COS

Das COS MUSS die MANAGE CHANNEL-Variante aus 14.9.8.3 unterstützen.

- b. K_COS, Option_logische_Kanäle
Das COS MUSS die MANAGE CHANNEL-Varianten aus 14.9.8.1, 14.9.8.2 und 14.9.8.4 unterstützen.
- c. Das COS KANN weitere MANAGE CHANNEL-Varianten
 - 1. unterstützen oder
 - 2. ablehnen.

(N099.545) K_COS

Zugriffsregeln für das MANAGE CHANNEL Kommando:

- a. Das COS KANN die Auswertung von Zugriffsregeln für das MANAGE CHANNEL Kommando unterstützen.
- b. Das COS KANN als Zugriffsbedingung für das MANAGE CHANNEL Kommando stets ALWAYS verwenden.

(N099.548) K_COS

Wenn *intendedAction* das

- a. Öffnen eines logischen Kanals (*intendedAction* = P1P2 = '0000') anzeigt
 - 1. und bereits alle verfügbaren logischen Kanäle geöffnet sind, dann MUSS das Kommando mit dem Trailer NoMoreChannelsAvailable terminieren,
 - 2. andernfalls MUSS das COS
 - i. eine nicht verwendete Kanalnummer *newChannelNumber* allokalieren, wobei die Zahl *newChannelNumber* aus dem Intervall [1, 19] gewählt werden MUSS.
 - ii. einen weiteren logischen Kanal öffnen,
 - iii. diesem die allokierte Nummer *newChannelNumber* zuweisen und
 - iv. dessen Kanalkontext entsprechend (N030.100) initialisieren.
 - 3. Als Datenfeld der Antwortnachricht MUSS I2OS(*newChannelNumber*, 1) verwendet werden.
- b. Schließen eines logischen Kanals (*intendedAction* = P1P2 = '8000') anzeigt
 - 1. dann MUSS der entsprechende logische Kanal geschlossen werden und
 - 2. die freiwerdende Kanalnummer MUSS für zukünftige Allokationen verfügbar sein.
 - 3. Das Datenfeld der Antwortnachricht MUSS leer sein.
- c. Zurücksetzen eines logischen Kanals (*intendedAction* = P1P2 = '4000') anzeigt,
 - 1. dann MUSS der *channelContext* des entsprechenden logischen Kanals auf den in (N030.100) definierten Wert gesetzt werden, und
 - 2. das Datenfeld der Antwortnachricht MUSS leer sein.
- d. logische Resetten (*intendedAction* = 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.

(N099.551) K_COS

Falls nicht anderweitig spezifiziert, MUSS als Trailer NoError gewählt werden.

(N099.554) K_COS

Für die Priorität der Trailer gilt:

- a. Die Priorität der Trailer in Tabelle 251 ist herstellerspezifisch.
- b. Jeder Trailer in Tabelle 251 MUSS eine höhere Priorität als NoError haben.

14.9.9 MANAGE SECURITY ENVIRONMENT

Das Kommando MANAGE SECURITY ENVIRONMENT (MSE) verändert im *currentFolder* die Attribute *selfIdentifier* 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.

(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*. *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:

- a. Der Wert *location* = '00' MUSS verwendet werden, wenn ein globaler Schlüssel betroffen ist.
- b. Der Wert *location* = '80' MUSS verwendet werden, wenn ein DF-spezifischer Schlüssel betroffen ist.
- c. Der Parameter *identifier* bestimmt das betroffene Schlüsselobjekt. Der Wert von *identifier* MUSS konform zu (N016.400) bzw. (N017.100) gewählt werden.
- d. 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:

(N099.700) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = 'F3' gewählt werden.

(N099.800) K_externeWelt {K_Karte}

Der Parameter *seNo* MUSS gemäß (N007.900) gewählt werden.

(N099.900) K_externeWelt {K_Karte}

Es MUSS eine Case 1 Kommando-APDU gemäß 11.7.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 1 Kommando-APDU MÜSSEN die Angaben aus Tabelle 252 verwendet werden.

Tabelle 252: 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'	<i>operationMode</i> = Auswahl eines SE-Identifiers
P2	'XX'	<i>seNo</i> = 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:

(N100.000) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '41' gewählt werden.

(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.

(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.

(N100.300) K_externeWelt {K_Karte}

- Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {
aesSessionkey4TC,
desSessionkey4TC (Option_DES),
} verwendet werden MUSS.
- Das COS KANN weitere Werte für *algId*
 - akzeptieren oder
 - ablehnen.

(N100.400) K_externeWelt {K_Karte}

Es MUSS eine Case 3 Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 253 verwendet werden.

Tabelle 253: 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'	<i>operationMode</i> = Setzen eines „internen“ Schlüssels
P2	'A4'	<i>crtTag</i> = betroffenes Listenelement ist internalAuthenticate
Data	'XX...XX'	'83 – L ₈₃ – keyRef 80 01 algId '

Hinweis (150): L₈₃ ist ein Oktett mit dem Wert l2OS(OctetLength(keyRef), 1).

14.9.9.3 Use Case Schlüsselauswahl zur internen, asymmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

(N100.500) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '41' gewählt werden.

(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.

(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.

(N100.800) K_externeWelt {K_Karte}

- a. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *internalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 oder Tabelle 270 gewählt werden, wobei ein Wert aus der Menge {
elcAsynchronAdmin,
elcRoleAuthentication,
elcSessionkey4SM,
elcSessionkey4TC,
rsaClientAuthentication,
rsaRoleAuthentication (Option_RSA_CVC),
rsaSessionkey4SM (Option_DES),
rsaSessionkey4TC (Option_DES),
signPKCS1_V1_5
} verwendet werden MUSS.
- b. Das COS KANN weitere Werte für *algId*
 1. akzeptieren oder
 2. ablehnen.

(N100.900) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 254 verwendet werden.

Tabelle 254: 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'	<i>operationMode</i> = Setzen eines „internen“ Schlüssels
P2	'A4'	<i>crtTag</i> = betroffenes Listenelement ist <i>internalAuthenticate</i>
Data	'XX...XX'	'84 01 <i>keyRef</i> 80 01 <i>algId</i> '

14.9.9.4 Use Case Schlüsselauswahl zur externen, symmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

(N101.000) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.

(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.

(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.

(N101.300) K_externeWelt {K_Karte}

- a. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {
aesSessionkey4TC,
desSessionkey4TC (Option_DES),
} verwendet werden MUSS.
- b. Das COS KANN weitere Werte für *algId*
 1. akzeptieren oder
 2. ablehnen.

(N101.400) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 255 verwendet werden.

Tabelle 255: 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'	<i>operationMode</i> = Setzen eines „externen“ Schlüssels <i>crtTag</i> = betroffenes Listenelement ist <i>externalAuthenticate</i>
P2	'A4'	
Data	'XX...XX'	'83 – L ₈₃ – <i>keyRef</i> 80 01 <i>algId</i> '

Hinweis (151): L₈₃ ist ein Oktett mit dem Wert *I2OS(OctetLength(keyRef), 1)*.

14.9.9.5 Use Case Schlüsselauswahl zur externen, asymmetrischen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

(N101.500) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.

(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.

(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) gewählt werden.

(N101.800) K_externeWelt {K_Karte}

- a. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {
 elcRoleCheck,
 rsaRoleCheck (Option_RSA_CVC),
 rsaSessionkey4SM (Option_DES),
 rsaSessionkey4TC (Option_DES)
} verwendet werden MUSS.
Hinweis: Die Schlüsselselektion für die Algorithmen elcSessionkey4SM und elcSessionkey4TC erfolgt im GENERAL AUTHENTICATE-Kommando, siehe 14.7.2.2.1.
- b. Das COS KANN weitere Werte für *algId*
1. akzeptieren oder
 2. ablehnen.

(N101.900) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 256 verwendet werden.

Tabelle 256: 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'	<i>operationMode</i> = Setzen eines öffentlichen Schlüssels
P2	'A4'	<i>crtTag</i> = betroffenes Listenelement ist <i>externalAuthenticate</i>
Data	'XX...XX'	'83-0C-keyRef 80-01- <i>algId</i> '

14.9.9.6 Use Case Schlüsselauswahl zur symmetrischen, gegenseitigen Authentisierung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

(N102.000) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.

(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.

(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.

(N102.300) K_externeWelt {K_Karte}

- a. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *externalAuthenticate*. Wert und Codierung MÜSSEN gemäß

Tabelle 268 gewählt werden, wobei ein Wert aus der Menge {
aesSessionkey4SM
desSessionkey4SM (Option_DES)
} verwendet werden MUSS.

- b. Das COS KANN weitere Werte für *algId*
 1. akzeptieren oder
 2. ablehnen.

(N102.400) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 257 verwendet werden.

Tabelle 257: 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'	<i>operationMode</i> = Setzen eines symmetrischen Schlüssels <i>crtTag</i> = betroffenes Listenelement ist <i>externalAuthenticate</i>
P2	'A4'	
Data	'XX...XX'	'83 01 <i>keyRef</i> 80 01 <i>algId</i> '

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.11.1] 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:

(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 Tabelle 271 gewählt werden. Falls die

- a. Option_kontaktlose_Schnittstelle unterstützt wird, dann MUSS für *OID* eine Wert aus der Menge {
 1. id-PACE-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
 2. id-PACE-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
 3. id-PACE-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)} verwendet werden.
- b. Option_PACE_PCD unterstützt wird, dann MUSS für *OID* ein Wert aus der Menge {
 1. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128, (siehe 15.4.2),
 2. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192, (siehe 15.4.2),
 3. id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256, (siehe 15.4.2)} verwendet werden.

(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.

(N102.448) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 258 verwendet werden.

Tabelle 258: MSE, Selektion sym. 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'	<i>operationMode</i> = Setzen eines geheimen Schlüsselobjektes <i>crtTag</i> = betr. Listenelemente: <i>externalAuthenticate</i> , <i>internalAuthenticate</i>
P2	'A4'	
Data	'XX...XX'	'80 I2OS(OctetLength(<i>OID</i>), 1) <i>OID</i> 83 01 <i>keyRef</i> '

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.11.1] 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:

(N102.450) K_externeWelt {K_Karte}

Der Parameter *OID* hat dieselbe Bedeutung und Codierung wie in (N102.440).

(N102.452) K_externeWelt {K_Karte}

Der Parameter *keyRef* hat dieselbe Bedeutung und Codierung wie in (N102.444).

(N102.454) K_externeWelt {K_Karte}

Der Parameter *idDomainParameter* enthält einen Identifier für Domainparameter der zu verwendenden elliptischen Kurve mit der in Tabelle 259 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 259: Zuordnung von *idDomainParameter* zu Domainparameter

ID	Domainparameter	<i>algorithmIdentifier</i> im sym. Kartenverbindungsobjekt
13 = '0D'	brainpoolP256r1	id-PACE-ECDH-GM-AES-CBC-CMAC-128 oder id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-128
16 = '10'	brainpoolP384r1	id-PACE-ECDH-GM-AES-CBC-CMAC-192 oder id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-192
17 = '11'	brainpoolP512r1	id-PACE-ECDH-GM-AES-CBC-CMAC-256 oder id-PACE-PCD-ECDH-GM-AES-CBC-CMAC-256

(N102.458) K_externeWelt {K_Karte}

Es MUSS eine Case 3S-Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle

„Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3-Kommando-APDU MÜSSEN die Angaben aus Tabelle 260 verwendet werden.

Tabelle 260: MSE, Selektion symmetrisches Kartenverbindungsobjekt

	Inhalt	Beschreibung
CLA	‘00’	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	‘22’	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	‘C1’	<i>operationMode</i> = Setzen eines geheimen Schlüsselobjektes <i>crtTag</i> = betr. Listenelemente: <i>externalAuthenticate</i> , <i>internalAuthenticate</i>
P2	‘A4’	
Data	‘XX...XX’	‘80 l2OS(OctetLength(<i>OID</i>), 1) <i>OID</i> 83 01 <i>keyRef</i> 84 01 <i>idDomainparameter</i> ‘

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 14.9.3) oder mit diesem Schlüssel Signaturen zu erzeugen (siehe 14.8.2). In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

(N102.500) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = ‘41’ gewählt werden.

(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.

(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.

(N102.800) K_externeWelt {K_Karte}

- Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *signatureCreation*. Wert und Codierung MÜSSEN gemäß Tabelle 268 oder Tabelle 270 gewählt werden, wobei ein Wert aus der Menge {
rsaClientAuthentication,
sign9796_2_DS2,
signECDSA,
signPKCS1_V1_5,
signPSS
} verwendet werden MUSS.
- Das COS KANN weitere Werte für *algId*
 - akzeptieren oder
 - ablehnen.

(N102.900) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 261 verwendet werden.

Tabelle 261: 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'	<i>operationMode</i> = Setzen eines privaten Schlüssels <i>crtTag</i> = betroffenes Listenelement ist signatureCreation
P2	'B6'	
Data	'XX...XX'	'84 01 <i>keyRef</i> 80 01 <i>algId</i> '

14.9.9.10 Use Case Schlüsselauswahl zum Prüfen von CV-Zertifikaten

In dieser Variante enthält die APDU des MSE Kommandos drei Parameter:

(N103.000) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.

(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.

(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) gewählt werden.

(N103.300) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 262 verwendet werden.

Tabelle 262: MSE, Selektion öffentlicher Zertifikatsprüfchlüssel

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'81'	<i>operationMode</i> = Setzen eines öffentlichen Schlüssels <i>crtTag</i> = betroffenes Listenelement ist verifyCertificate
P2	'B6'	
Data	'XX...XX'	'83 08 <i>keyRef</i> '

14.9.9.11 Use Case Schlüsselauswahl zur Datenent- oder Datenumschlüsselung

In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

(N103.400) K_externeWelt {K_Karte}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '41' gewählt werden.

(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.

(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.

(N103.700) K_externeWelt {K_Karte}

- a. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *dataDecipher*. Wert und Codierung MÜSSEN gemäß Tabelle 269 gewählt werden, wobei ein Wert aus der Menge {
 elcSharedSecretCalculation,
 rsaDecipherPKCS1_V1_5,
 rsaDecipherOaep
} verwendet werden MUSS.
- b. Das COS KANN weitere Werte für *algId*
 1. akzeptieren oder
 2. ablehnen.

(N103.800) K_externeWelt {K_Karte}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 263 verwendet werden.

Tabelle 263: MSE, Schlüsselselektion zur Entschlüsselung

	Inhalt	Beschreibung
CLA	'00'	CLA-Byte gemäß [ISO/IEC 7816-4]
INS	'22'	Instruction Byte gemäß [ISO/IEC 7816-4]
P1	'41'	<i>operationMode</i> = Setzen eines privaten Schlüssels <i>crtTag</i> = betroffenes Listenelement ist <i>dataDecipher</i>
P2	'B8'	
Data	'XX...XX'	'84 01 <i>keyRef</i> 80 01 <i>algId</i> '

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 14.8.4.3 und 14.8.4.4). In dieser Variante enthält die APDU des MSE Kommandos vier Parameter:

(N103.830) K_externeWelt {K_COS}

Der Parameter *operationMode* bestimmt die durchzuführende Aktion. Für diesen Use Case MUSS *operationMode* = '81' gewählt werden.

(N103.835) K_externeWelt {K_COS}

Der Parameter *crtTag* bestimmt das Listenelement in *keyReferenceList*, welches zu ändern ist. Für diesen Use Case MUSS *crtTag* = 'B8' gewählt werden.

(N103.840) K_externeWelt {K_COS}

Der Parameter *keyRef* enthält den neuen Wert für das Element *keyReference* im Listenelement *dataEncipher*. Wert und Codierung MÜSSEN gemäß (N019.822) gewählt werden.

(N103.845) K_externeWelt {K_COS}

- a. Der Parameter *algId* enthält den neuen Wert für das Element *algorithmIdentifier* im Listenelement *dataEncipher*. Wert und Codierung MÜSSEN gemäß Tabelle

269 gewählt werden, wobei ein Wert aus der Menge {
elcSharedSecretCalculation,
rsaEncipherOaep,
rsaEncipherPKCS1_V1_5
} verwendet werden MUSS.

- b. Das COS KANN weitere Werte für *algId*
1. akzeptieren oder
 2. ablehnen.

(N103.850) K_externeWelt {K_COS}

Es MUSS eine Case 3S Kommando-APDU gemäß 11.7.3.1 über die Schnittstelle „Interpreter“ in Abbildung 1 geschickt werden. Für die Konstruktion dieser Case 3 Kommando-APDU MÜSSEN die Angaben aus Tabelle 261 verwendet werden.

Tabelle 264: 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'	<i>operationMode</i> = Setzen eines öffentlichen Schlüssels <i>crtTag</i> = betroffenes Listenelement ist dataEncipher
P2	'B8'	
Data	'XX...XX'	'83 0C <i>keyRef</i> 80 01 <i>algId</i> '

14.9.9.13 Antwort der Karte auf Management des Security Environments

Tabelle 265: MSE Antwort-APDU im Erfolgsfall

Trailer	Inhalt	Beschreibung
'90 00'	NoError	Erfolgreiche Übernahme der Kommandodatenparameter

Tabelle 266: 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 (152): Diese Tabelle enthält keine Fehler, die in den Komponenten I/O, ChannelSwitch und SecMes aus Abbildung 1 entdeckt wurden.

(N103.900) K_COS

Ein COS KANN zusätzliche Trailer verwenden.

14.9.9.14 Kommandoabarbeitung innerhalb der Karte

(N104.000) Kommandounterstützung

- a. K_COS
Das COS MUSS die Manage Security Environment-Varianten aus 14.9.9.1, 14.9.9.3, 14.9.9.5, 14.9.9.6, 14.9.9.9, 14.9.9.10 und 14.9.9.11 unterstützen.
- b. K_COS, Option_Kryptobox
Das COS MUSS die MANAGE SECURITY ENVIRONMENT-Varianten aus 14.9.9.2, 14.9.9.4 und 14.9.9.12 unterstützen.
- c. K_COS, Option_kontaktlose_Schnittstelle
Das COS MUSS die MANAGE SECURITY ENVIRONMENT-Varianten aus 14.9.9.7 und 14.9.9.8 unterstützen.
- d. K_COS, Option_PACE_PCD
Das COS MUSS die MANAGE SECURITY ENVIRONMENT-Variante aus 14.9.9.7 und 14.9.9.8 unterstützen.
- e. Das COS KANN weitere MANAGE SECURITY ENVIRONMENT-Varianten
 - 1. unterstützen oder
 - 2. ablehnen.

(N104.100) K_COS

Zugriffsregeln für das MANAGE SECURITY ENVIRONMENT Kommando:

- a. Das COS KANN die Auswertung von Zugriffsregeln für das MANAGE SECURITY ENVIRONMENT Kommando unterstützen.
- b. Das COS KANN als Zugriffsbedingung für das MANAGE SECURITY ENVIRONMENT Kommando stets ALWAYS verwenden.

(N104.200) K_COS

Wenn der Parameter *operationMode* in P1 den Wert 'F3' besitzt, dann MUSS

- a. in *currentFolder* das Attribut *seldentifier* (siehe (N030.000)a) auf den Wert des Parameters *seNo* gesetzt werden.
- b. jedes Listenelement in *keyReferenceList* (siehe (N029.900)c), welches auf ein Schlüsselobjekt in *currentFolder* zeigt, auf den Wert „leer“ gesetzt werden.
- c. jedes Element in *globalPasswordList* (siehe (N029.900)i) und in *dfSpecificPasswordList* (siehe (N029.900)j) mittels *clearPasswordStatus(...)* aus den genannten Listen entfernt werden, falls es *currentFolder* zugeordnet ist.
- d. der Wert von *currentEF* unverändert bleiben.
- e. *clearSecurityStatusFolder(currentFolder)* ausgeführt werden.
- f. Attribute, die übergeordneten Ordnern zugeordnet sind, DÜRFEN sich NICHT ändern.

(N104.300) K_COS

Wenn der Parameter *operationMode* in P1 einen Wert aus der Menge {'41', '81', 'C1'} besitzt und

- a. im Falle einer ein Oktett langen Schlüsselreferenz *keyRef*
 - 1. SOLL entweder mittels *SearchKey(channelContext.currentFolder, keyRef, algId)* nach dem Schlüssel gesucht werden. Wenn 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.
- 2. oder KANN keine Schlüsselsuche durchgeführt und als Trailer `NoError` gewählt werden.
- b. im Falle einer acht Oktett (öffentliches Signaturprüfobjekt) langen Schlüsselreferenz, MUSS `keyRef` mittels `SearchKey(channelContext.currentFolder, keyRef, verifyCertificate)` nach dem Schlüssel gesucht werden. Wenn diese Funktion
 - 1. den Fehler `keyNotFound` meldet, dann MUSS das Kommando mit dem Trailer `KeyNotFound` terminieren.
 - 2. den Fehler `notSupported` meldet, dann MUSS das Kommando mit dem Trailer `UnsupportedFunction` terminieren.
 - 3. keinen Fehler meldet, dann MUSS als Trailer `NoError` gewählt werden.
- c. im Falle einer zwölf Oktett (öffentliches Authentisierungsobjekt, öffentliches Verschlüsselungsobjekt) langen Schlüsselreferenz, MUSS `keyRef` mittels `SearchKey(channelContext.currentFolder, keyRef, algId)` nach dem Schlüssel gesucht werden. Wenn diese Funktion
 - 1. den Fehler `keyNotFound` meldet, dann MUSS das Kommando mit dem Trailer `KeyNotFound` terminieren.
 - 2. den Fehler `notSupported` meldet, dann MUSS das Kommando mit dem Trailer `UnsupportedFunction` terminieren.
 - 3. keinen Fehler meldet, dann MUSS als Trailer `NoError` gewählt werden.

(N104.320) K_COS

Falls als Trailer

- a. `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.
- b. 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.

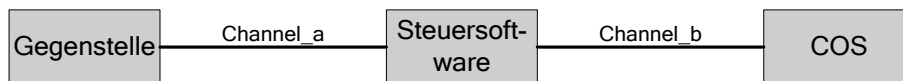


Abbildung 3: Kommunikationsmodell Gegenstelle, Steuersoftware und COS

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 Abbildung 1.

Jedem, der hier behandelten Verfahren, ist ein eigenes Unterkapitel gewidmet. Im Allgemeinen besteht die Authentisierung aus einer Sequenz von einem oder mehreren Kommandos, die über die Kommunikationskanäle gesendet werden. Für alle Authentisierungsprotokolle gilt:

(N104.600) K_externeWelt {K_Karte}

Falls 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 Abbildung 3) NICHT durch Kommandos unterbrochen werden, welche nicht zu dieser Sequenz gehören.

(N104.700) K_COS

Falls die Anforderung (N104.600) von der externen Entität nicht eingehalten wird, mithin also die Sequenz durch ein unterbrechendes Kommando unterbrochen wird, dann KANN das COS

- a. das unterbrechende Kommando akzeptieren.
- b. das unterbrechende Kommando ablehnen.
- c. ein Fortsetzen der unterbrochenen Sequenz akzeptieren.
- d. ein Fortsetzen der unterbrochenen Sequenz ablehnen.

(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 Kapitel vermerkt.

(N104.850) K_externeWelt {K_Karte}

Wenn das erste Kommando dieser Sequenz

- geschützt übertragen wird, dann MÜSSEN auch alle anderen Kommandos dieser Sequenz geschützt übertragen werden.
- ungeschützt übertragen wird, dann MÜSSEN auch alle anderen Kommandos dieser Sequenz ungeschützt übertragen werden.

(N104.900) K_COS

Falls die Anforderung (N104.800) von der externen Entität nicht eingehalten wird, dann KANN das COS dies

- akzeptieren oder
- ablehnen.

15.1 Externe Authentisierung

Dieses Kapitel behandelt die Authentisierung einer „Gegenstelle“ gegenüber dem COS.

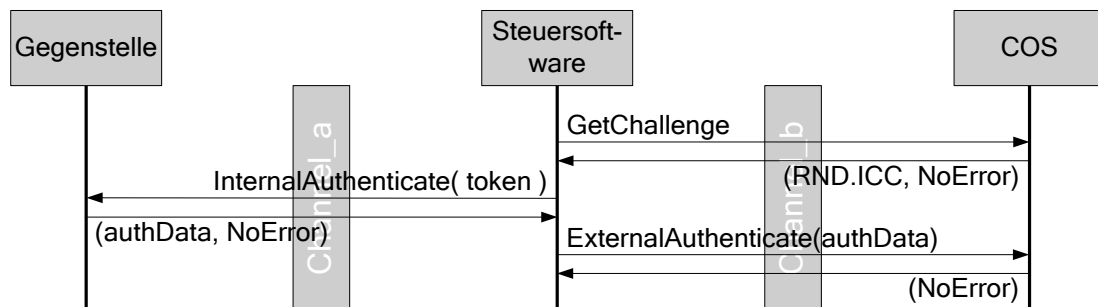


Abbildung 4: Sequenzdiagramm zur externen Authentisierung

Hinweis (153): Die Bedeutung der Bezeichnungen in Abbildung 4 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) Diese Anforderung ist absichtlich leer.

(N105.100) Diese Anforderung ist absichtlich leer.

(N105.200) Diese Anforderung ist absichtlich leer.

15.1.2 RSA, asymmetrische Rollenauthentisierung, Option_RSA_CVC

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 Abbildung 4):

(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.

(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.

(N105.500) K_COS

Es MUSS möglich sein, die Kommandos dieser Sequenz, welche über Channel_b gesendet werden, geschützt (mit Secure Messaging) zu übertragen.

15.1.3 ELC, asymmetrische 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 Abbildung 4):

(N105.600) K_externeWelt {K_Karte}

Das erste Kommando der Sequenz MUSS GET CHALLENGE gemäß 14.9.4.2 sein und über Channel_b geschickt werden.

(N105.700) K_externeWelt {K_Karte}

Das zweite Kommando über Channel_b ist das letzte dieser Sequenz und MUSS ein EXTERNAL AUTHENTICATE gemäß 14.7.1.1 mit *algId* aus der Menge {*elcRoleCheck*} sein.

(N105.800) K_COS

Es MUSS möglich sein, die Kommandos dieser Sequenz, welche über Channel_b gesendet werden, geschützt (mit Secure Messaging) zu übertragen.

15.2 Interne Authentisierung

Für den Fall, dass eine externe Entität die Authentizität des COS mittels eines Schlüssels überprüfen will und dabei keine Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 5):

(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 14.7.4.1 mit einer *algId* aus folgender Menge sein: {

elcRoleAuthentication,
rsaClientAuthentication,
rsaRoleAuthentication (Option_RSA_CVC),
signPKCS1_V1_5

}.

(N106.000) K_COS

Es MUSS möglich sein, die Kommandos dieser Sequenz, welche über Channel_b gesendet werden, geschützt (mit Secure Messaging) zu übertragen.

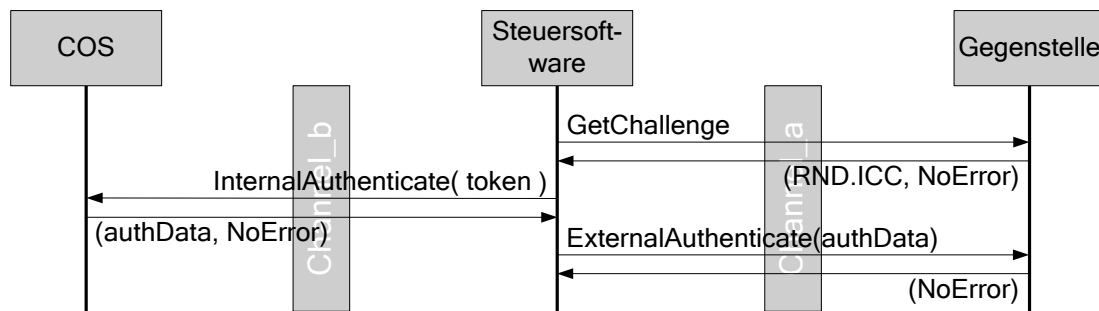


Abbildung 5: Sequenzdiagramm zur internen Authentisierung

Hinweis (154): Die Bedeutung der Bezeichnungen in Abbildung 5 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 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, Gegenstelle und 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 Abbildung 6):

(N106.100) K_externeWelt {K_Karte}

Das erste Kommando der Sequenz MUSS GET CHALLENGE gemäß 14.9.4.1 sein und über Channel_b geschickt werden. Die dabei vom COSb erzeugte Zufallszahl wird mit RND.1 bezeichnet.

(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 14.7.4.1 mit *algId* gleich *symSessionkey4TC* sein. Dabei MUSS *cmdData* RND.1 und *iccsn8* des COSb gemäß M2 aus (N084.400)b.1 enthalten. Die Antwortdaten des COSa werden mit *rspData.2* bezeichnet.

(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 14.7.1.2 mit *algId* gleich symSessionkey4SM sein. Als Kommandodaten MÜSSEN rspData.2 verwendet werden. Die Antwortdaten des COSb werden mit rspData.3 bezeichnet.

(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 14.7.1.1 mit *algId* gleich symSessionkey4TC sein. Als Kommandodaten MÜSSEN rspData.3 verwendet werden.

(N106.500) K_COS

Sowohl COSa als auch COSb KANN die geschützte Übertragung der Sequenz

- erlauben oder
- ablehnen.

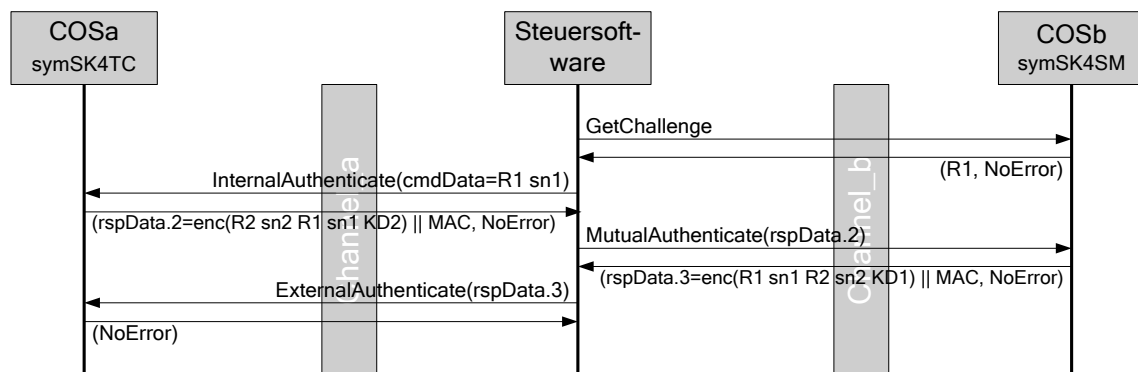


Abbildung 6: Sequenzdiagramm symmetrische Sessionkey-Aushandlung

Hinweis (155): 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 Abbildung 7):

(N106.540) K_COS, Option_kontaktlose_Schnittstelle

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:

- id-PACE-ECDH-GM-AES-CBC-CMAC-128 mit brainpoolP256r1
- id-PACE-ECDH-GM-AES-CBC-CMAC-192 mit brainpoolP384r1
- id-PACE-ECDH-GM-AES-CBC-CMAC-256 mit brainpoolP512r1

(N106.545) K_COS

Verfügbarkeit PACE Authentisierungsprotokoll:

- Option_kontaktlose_Schnittstelle: Das COS MUSS die PACE-Varianten aus (N106.540) in der Rolle „MRTD Chip (PICC)“ unterstützen.

- b. Option_PACE_PCD: Das COS MUSS die PACE-Varianten aus (N106.540) in der Rolle „Terminal (PCD)“ unterstützen.

Hinweis (156): In (N106.545)b ist absichtlich nur die Option_PACE_PCD enthalten, nicht aber die Option_kontaktlose_Schnittstelle.

(N106.547) K_COS

Das COS KANN die geschützte Übertragung der Sequenz

- a. erlauben oder
- b. ablehnen.

(N106.550) K_externeWelt {K_Karte}

Im ersten Schritt MUSS zur Komponente

- a. COSa ein GENERAL AUTHENTICATE-Kommando gemäß 14.7.2.1.1 geschickt werden. Die Antwortdaten enthalten das Kryptogramm z , welches im zweiten Schritt auf Seiten der Komponente COSb benötigt wird.
- b. COSb ein GENERAL AUTHENTICATE-Kommando gemäß 14.7.2.4.1 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK1_{PCD}$, der im zweiten Schritt auf Seiten der Komponente COSa benötigt wird.

(N106.552) K_externeWelt {K_Karte}

Im zweiten Schritt MUSS zur Komponente

- a. COSa ein GENERAL AUTHENTICATE-Kommando mit $\sim PK1_{PCD}$ aus (N106.550)b gemäß 14.7.2.1.2 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK1_{PICC}$ welcher im dritten Schritt auf Seiten der Komponente COSb benötigt wird.
- b. COSb ein GENERAL AUTHENTICATE-Kommando mit z aus (N106.550)a gemäß 14.7.2.4.2 geschickt werden. Die Antwortnachricht enthält kein Datenfeld.

(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 14.7.2.4.3 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK2_{PCD}$ welcher im vierten Schritt auf Seiten der Komponente COSa benötigt wird.

(N106.556) K_externeWelt {K_Karte}

Im vierten Schritt MUSS zum COSa ein GENERAL AUTHENTICATE-Kommando mit $\sim PK2_{PCD}$ aus (N106.554) gemäß 14.7.2.1.3 geschickt werden. Die Antwortdaten enthalten den ephemeren öffentlichen Schlüssel $\sim PK2_{PICC}$ welcher im fünften Schritt auf Seiten der Komponente COSb benötigt wird.

(N106.558) K_externeWelt {K_Karte}

Im fünften Schritt MUSS zum COSb ein GENERAL AUTHENTICATE-Kommando mit $\sim PK2_{PICC}$ aus (N106.556) gemäß 14.7.2.4.4 geschickt werden. Die Antwortdaten enthalten einen CMAC T_{PCD} über $\sim PK2_{PICC}$ welcher im sechsten Schritt auf Seiten der Komponente COSa benötigt wird.

(N106.560) K_externeWelt {K_Karte}

Im sechsten Schritt MUSS zum COSa ein GENERAL AUTHENTICATE-Kommando mit T_{PCD} aus (N106.558) gemäß 14.7.2.1.4 geschickt werden. Die Antwortdaten enthalten einen CMAC T_{PICC} über $\sim PK2_{PCD}$ welcher im siebten Schritt auf Seiten der Komponente COSb benötigt wird.

(N106.562) K_externeWelt {K_Karte}

Im siebten Schritt MUSS zum COSb ein GENERAL AUTHENTICATE-Kommando mit

T_{PICC} aus (N106.560) gemäß 14.7.2.4.5 geschickt werden. Die Antwortnachricht enthält kein Datenfeld.

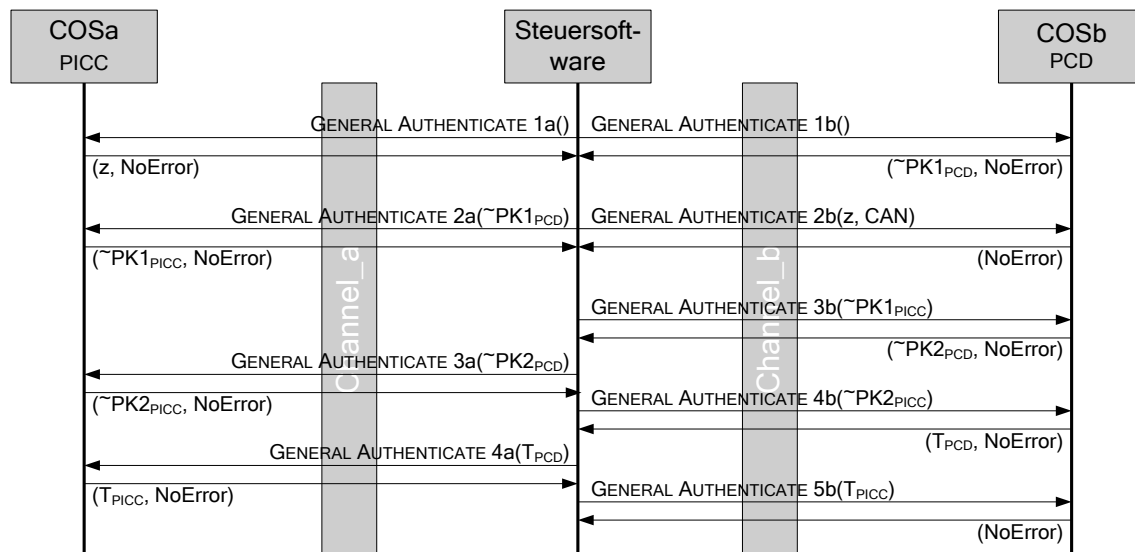


Abbildung 7: Sequenzdiagramm PACE Authentisierung

Hinweis (157): Aus Gründen der Übersichtlichkeit wurden in Abbildung 7 kürzere Bezeichnungen verwendet, als bei der Beschreibung der Kommandos in Kapitel 14. Die Bedeutung der Bezeichnungen ergibt sich aus dem jeweiligen Kontext.

Hinweis (158): 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

Für den Fall, dass eine gegenseitige Authentisierung mittels asymmetrischer RSA-Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 8):

(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.

(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 `algld` 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.

(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 `algld` gleich `rsaSessionkey4TC` sein. Als Kommandodaten MÜSSEN `rspData.2` verwendet werden.

(N106.900) K_externeWelt {K_Karte}, Option_DES

Das vierte Kommando MUSS zum COSb gesendet werden und MUSS GET CHAL-

LENGE gemäß 14.9.4.1 sein. Die dabei vom COSb erzeugten Antwortdaten werden mit `rspData.4` bezeichnet.

(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 `algld` gleich `rsaSessionkey4TC` sein. Dabei MUSS `cmdData.5 = rspData.4 || ICCSN8.COS` gelten mit `ICCSN8.COS` gleich `ic-csn8` (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.

(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 EXTERNAL AUTHENTICATE gemäß 14.7.1.1 mit `algld` gleich `rsaSessionkey4SM` sein. Als Kommandodaten MÜSSEN `rspData.5` verwendet werden.

(N107.200) `K_COS_G1`, Option_DES

Sowohl COSa als auch COSb KANN die geschützte Übertragung der Sequenz

- a. erlauben oder
- b. ablehnen.

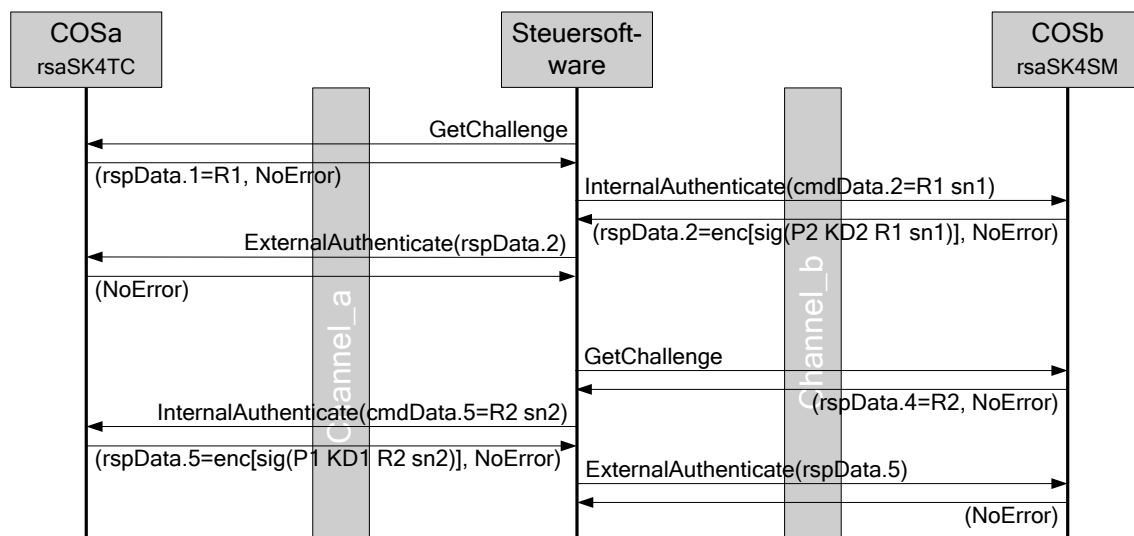


Abbildung 8: Sequenzdiagramm RSA-Sessionkey-Aushandlung

Hinweis (159): Aus Gründen der Übersichtlichkeit wurden in Abbildung 8 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 (160): 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 Verwendungszwe-

cke 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 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 $K1 = ECKA(PrK.a, PK.b, D)$ berechnet.

- iii) das zweite gemeinsame Geheimnis $K2 = ECKA(SK.a, PuK.b, D)$ berechnet.
 - iv) Komponente *A* bildet die Schlüsselableitungsdaten $K = K1 \parallel K2$.
 - 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 $K1 = ECKA(SK.b, PuK.a, D)$ berechnet.
 - iii) das zweite gemeinsame Geheimnis $K2 = ECKA(PrK.b, PK.a, D)$ berechnet.
 - iv) Komponente *B* bildet die Schlüsselableitungsdaten $K = K1 \parallel K2$.
 - 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 könnte der Angreifer dann mittels PSO Encipher beliebige Paare von „plaintext“ und „ciphertext“ erzeugen 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.

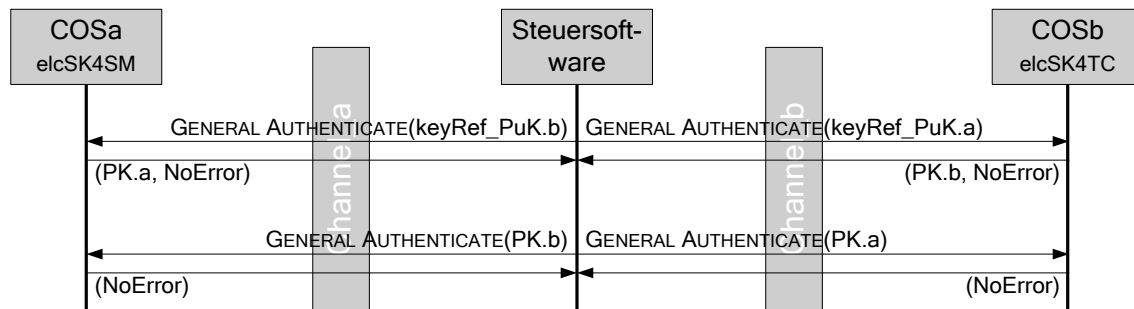


Abbildung 9: Sequenzdiagramm ELC-Sessionkey-Aushandlung

Für den Fall, dass eine gegenseitige Authentisierung mittels asymmetrischer ELC-Schlüssel durchzuführen ist und dabei Sessionkeys ausgehandelt werden, ist folgende Sequenz zu wählen (vergleiche Abbildung 9):

(N107.220) *K_externeWelt* {*K_Karte*}

Im ersten Schritt MUSS zu jeder der beiden Komponenten ein GENERAL AUTHENTICATE-Kommando gemäß 14.7.2.2.1 geschickt werden. Die beiden Antwortdaten enthalten einen ephemeren, öffentlichen Schlüssel.

(N107.230) *K_externeWelt* {*K_Karte*}

Im zweiten und letzten Schritt dieser Sequenz MUSS zu jeder der beiden Komponente ein GENERAL AUTHENTICATE-Kommando gemäß 14.7.2.2.2 geschickt werden.

(N107.235) *K_COS*

Sowohl COSa als auch COSb KANN die geschützte Übertragung der Sequenz

- erlauben oder
- ablehnen.

Hinweis (161): Aus Performanzgründen ist es für die Steuer-ware 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.

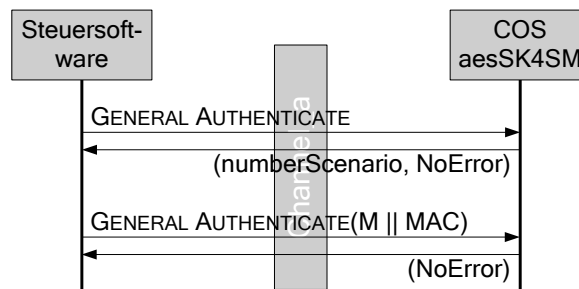


Abbildung 10: Sequenzdiagramm zur Übertragung von Sessionkeys

Hinweis (162): Die Bedeutung der Bezeichnungen in Abbildung 10 ergibt sich aus dem jeweiligen Kontext der Kommandos gemäß Kapitel 14.

Für den Fall, dass eine externe Entität Sessionkeys übertragen will, ist folgende Sequenz zu wählen (vergleiche Abbildung 10):

(N107.250) K_externeWelt {K_Karte}

Das erste Kommando der Sequenz MUSS über Channel_a geschickt werden und ein GENERAL AUTHENTICATE Kommando

- a. gemäß 14.7.2.3.1 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.

(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 14.7.2.3.2 sein, falls der in (N085.051)b beschriebene Fall vorliegt, oder
- b. gemäß 14.7.2.5.2 sein, falls der in (N085.051)d beschriebene Fall vorliegt.

(N107.254) K_COS

Das COS KANN die geschützte Übertragung

- a. erlauben oder
- b. ablehnen.

Hinweis (163): Hintergründe zu dieser Art der Sessionkeyübertragung finden sich in Anhang F.

16 Verschiedenes (normativ)

16.1 Identifier

Tabelle 267: Generische Algorithm-Identifier 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 268: Konkrete Algorithm-Identifier 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 269: AlgorithmIdentifier für Ver- und Entschlüsselung

Name	Codierung	Verwendung
aesSessionkey	0000 0000 ₂ = '00'	PSO Decipher + PSO Encipher
desSessionkey, Option_DES		
rsaDecipherOaep	1000 0101 ₂ = '85'	PSO Decipher mit RSA
rsaDecipherPKCS1_V1_5	1000 0001 ₂ = '81'	
rsaEncipherOaep	0000 0101 ₂ = '05'	PSO Encipher + Transcipher mit RSA, gleich AlgorithmIdentifier für Decipher mod 128
rsaEncipherPKCS1_V1_5	0000 0001 ₂ = '01'	
elcSharedSecretCalculation	0000 1011 ₂ = '0B'	PSO Decipher mit ELC

Tabelle 270: AlgorithmIdentifier für Integrität und Authentizität

Name	Codierung	Verwendung
aesSessionkey	0000 0000 ₂ = '00'	PSO Compute Cryptographic Checksum + 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 271: 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

Name und Codierung	Bemerkung
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] 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] 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] 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`	[BSI-TR-03110-3] 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`	[BSI-TR-03110-3] 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`	[BSI-TR-03110-3] 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} `2A8214004C048119`	Werte gemäß [gemSpec_OID#Tab_PKI_408], Interpretation der <i>flagList</i> 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 <i>flagList</i> gemäß [gemSpec_PKI#Tab_PKI_910]
rsaEncryption {1.2.840.113549.1.1.1} `2A864886F70d010101`	Verschlüsselung gemäß [PKCS#1] Kapitel 7.2.1
secp256r1	identisch zu ansix9p256r1, siehe dort
secp384r1	identisch zu ansix9p384r1, siehe dort

Name und Codierung	Bemerkung
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 272: 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 Fehlbedienungs Zählers
`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 Fehlbedienungs Zählers nicht mehr möglich
`69 83`	KeyExpired	Der Gültigkeitsbereich des Schlüssels ist abgelaufen
`69 83`	PasswordBlocked	Fehlbedienungs Zähler 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 REF. 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	<i>newFileSize</i> kein Vielfaches der Rekordlänge
`6A 80`	NumberPreconditionWrong	Vorbedingung zum Laden des Szenarios nicht erfüllt

Wert	Name	Bedeutung
'6A 80'	NumberScenarioWrong	Scenario wurde bereits geladen
'6A 80'	VerificationError	Fehlerhaftes CV-Zertifikat
'6A 80'	WrongCiphertext	Fehlerhaftes Chiffre
'6A 80'	WrongToken	Token ist fehlerhaft
'6A 81'	UnsupportedFunction	Schlüssel unterstützt den angegebenen 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

Anhang A – 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) 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#3.4] geforderte Qualität für K3-DRNG oder DRG.2 oder PTG.2 aufweisen.
3. Dieses Dokument beschreibt in 14.9.2 lediglich die funktionalen Anforderungen an die Fingerprintberechnung. 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 über Seitenkanäle die Smartcard verlassen.

Anhang B – Vorgaben zur Performanz

B.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.

B.2 Messaufbau (normativ)

Die Bearbeitungszeiten eines Kommandos werden mit dem im Folgenden beschriebenen Messaufbau durchgeführt:



Abbildung 11: 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 B.4).

Die Komponente „**Device under Test**“ (**DuT**) repräsentiert den Prüfling, dessen Performanz zu ermitteln ist.

B.3 Anforderungen an die Steuersoftware (normativ)

(N107.300) K_externeWelt {K_Karte}

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.

B.4 Anforderungen an das Interface Device (IFD) (normativ)

(N107.350) K_externeWelt {K_Karte}

Bezüglich der Schnittstelle des Interface Devices IFD zum Prüfling DuT gilt:

- Das IFD MUSS das Übertragungsprotokoll T=1 gemäß 11.2.1 unterstützen.
- Das IFD SOLL das Übertragungsprotokoll [ISO/IEC 7816-12] gemäß 11.2.2 unterstützen.
- Das IFD SOLL das kontaktlose Übertragungsprotokoll gemäß 11.2.3 unterstützen.

B.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.

(N107.400) K_externeWelt {K_Karte}

Der Übertragungskanal TPDU_Channel des IFD SOLL die maximale Übertragungskapazität des Prüflings DuT unterstützen. Im Einzelnen bedeutet das:

(N107.500) K_externeWelt {K_Karte}

Das IFD MUSS im Rahmen einer PPS-Sequenz alle Werte für PPS1 aus der Menge {'18', '95', '96', '97'} unterstützen.

Tabelle 273: 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

(N107.600) K_externeWelt {K_Karte}

Das IFD MUSS eine Versorgungsspannung U_{cc} gemäß [ISO/IEC 7816-3#Table 1] für Class A und Class B liefern.

(N107.700) K_externeWelt {K_Karte}

Das IFD MUSS einen Versorgungsstrom I_{cc} gemäß [ISO/IEC 7816-3#Table 1] für Class A und Class B liefern.

(N107.800) K_externeWelt {K_Karte}

Das IFD MUSS während der Performanzmessung ein Clocksignal mit einer Frequenz aus dem Intervall [4,95, 5,05] MHz liefern.

- (N107.900) K_externeWelt {K_Karte}
Das IFD MUSS „direct convention“ (siehe [ISO/IEC 7816-3#8.1]) unterstützen.
- (N108.000) K_externeWelt {K_Karte}
Falls das Byte TC₁ im ATR fehlt, dann MUSS das IFD beim Senden eine „extra guard time“ von 12 etu verwenden (siehe [ISO/IEC 7816-3#8.3]).
- (N108.100) K_externeWelt {K_Karte}
Falls das Byte TC₁ 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]).
- (N108.200) K_externeWelt {K_Karte}
Das IFD MUSS beim Empfang eine „character guard time“ von 11 etu unterstützen (siehe [ISO/IEC 7816-3#11.2]).
- (N108.300) K_externeWelt {K_Karte}
Das IFD MUSS BGT = 22 etu verwenden (siehe [ISO/IEC 7816-3#11.2]).
- (N108.400) K_externeWelt {K_Karte}
Das IFD MUSS für IFSC einen Wert von 254 unterstützen (siehe [ISO/IEC 7816-3#11.4.2]).
- (N108.500) K_externeWelt {K_Karte}
Das IFD MUSS für IFSD einen Wert von 254 unterstützen (siehe [ISO/IEC 7816-3#11.4.2]).
- (N108.600) K_externeWelt {K_Karte}
Das IFD
- MUSS für eine Kommandonachricht 4096 und SOLL 32.768 als APDU Länge unterstützen.
 - MUSS für eine Antwortnachricht 65.536 als APDU Länge unterstützen.

B.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 (164): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

B.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 (165): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

B.5 Allgemeines (normativ)

B.5.1 Normale Zeitmessung

B.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.

(N108.700) K_externeWelt {K_Karte}

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:
 1. 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 redundancy code“ enthält (siehe [ISO/IEC 7816-3#11.3.4]):
$$N_{cmd_TPDU} = N_{cmd_APDU} + 8 \text{ ceiling}(N_{cmd_APDU} / 254) - 4.$$
 2. 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]):
$$N_{cmd_etu} = 11 N_{cmd_TPDU} + 22 \text{ ceiling}(N_{cmd_APDU} / 254).$$
 3. Analog gilt für die Antwort-APDU:
$$N_{rsp_TPDU} = N_{rsp_APDU} + 8 \text{ ceiling}(N_{rsp_APDU} / 254) - 4$$
und
$$N_{rsp_etu} = 11 N_{rsp_TPDU} + 22 \text{ floor}(N_{rsp_APDU} / 254).$$
 4. Daraus folgt: $t_{IO} = (N_{cmd_etu} + N_{rsp_etu}) \text{ Teilerfaktor} / f_{max}$ mit Teilerfaktor und f_{max} in Abhängigkeit von PPS1 aus Tabelle 273.

B.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 (166): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

B.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 (167): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

B.5.2 Reguläre Aktivierung der Smartcard

B.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.

(N108.800) K_externeWelt {K_Karte}

Schritt 1: Im Rahmen der regulären Aktivierung der Smartcard MUSS zunächst eine Aktivierung gemäß (N023.920)a und (N023.920)b erfolgen.

(N108.900) K_externeWelt {K_Karte}

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 TA₁ aus dem ATR verwendet werden.

(N109.000) K_externeWelt {K_Karte}

Schritt 3: Das IFD MUSS der Smartcard den Wert von IFSD = 254 präsentieren.

(N109.100) K_externeWelt {K_Karte}

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 14.2.6.1 verwendet werden.

Hinweis (168): 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.

B.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 (169): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

B.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 (170): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

B.5.3 Punkteermittlung

Input:	t	n-Tupel mit Zeiten von n Einzelmessungen
	T _R	Referenzzeit
Output:	P	Gewichtete Qualität der Messzeit
Errors:	–	Keine
Notation:		P = points(t, T _R)

Es gelten folgende Definitionen:

\bar{X} = Mittelwert (Erwartungswert) aller Einzelmessungen im n -Tupel \underline{t} .

σ = Standardabweichung aller Einzelmessungen im n -Tupel \underline{t} .

(N109.200) K_externeWelt {K_Karte}

Es gilt $f_1 = e^{-\frac{\sigma}{\bar{X}}}$.

(N109.300) $K_externeWelt \{K_Karte\}$

$$\text{Es gilt} \quad f_2 = 1 - \left(\frac{X}{2T_R} \right)^2.$$

(N109.400) $K_externeWelt \{K_Karte\}$

$$\text{Es gilt} \quad P = f_1 \cdot f_2 \cdot T_R.$$

Die Funktion P besteht aus den Faktoren f_1 und f_2 und wird durch T_R gewichtet.

Falls die Standardabweichung σ 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.

Falls der Mittelwert X der Messergebnisse gleich dem Referenzwert T_R ist, dann ist der Faktor f_2 gleich 0,75.

Das Produkt aus f_1 und 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 Prüfpunktes im Rahmen zusammengesetzter Kommandosequenzen.

B.5.4 Gesamtbewertung

Die Gesamtbewertung wird durch gewichtetes Addieren der Einzelpunktzahlen ermittelt. Die Gewichte korrelieren mit der Häufigkeit des Auftretens der Einzelpositionen im Feld.

Tabelle 274: Gesamtbewertung für das Basisbetriebssystem

Prüfpunkt				$T_{Ri} / [\text{ms}]$	g_i	$g_i T_{Ri} / [\text{ms}]$
Kanalkapazität	(N024.100)	B.6.1	P_{IO}	17	2.000	34.000
Karte starten	(N023.920)	B.7	$P_{KarteStarten}$	60	500	30.000
ACTIVATE Datei	14.2.1.1	B.8.1	$P_{activate_EF}$	30	1	30
ACTIVATE Ordner			$P_{activate_DF}$	30	1	30
ACTIVATE private Key	14.2.1.2		$P_{activate_PrK}$	30	1	30
ACTIVATE sym. Key			$P_{activate_SK}$	30	1	30
ACTIVATE PuK	14.2.1.3		$P_{activate_PuK}$	30	1	30
ACTIVATE Pwd	14.2.1.4		$P_{activate_Pwd}$	30	1	30
DEACTIVATE Datei	14.2.3.1		$P_{deactivate_EF}$	30	1	30
DEACTIVATE Ordner			$P_{deactivate_DF}$	30	1	30
DEACTIVATE private Key	14.2.3.2		$P_{deactivate_PrK}$	30	1	30
DEACTIVATE sym. Key			$P_{deactivate_SK}$	30	1	30
DEACTIVATE PuK	14.2.3.3		$P_{deactivate_PuK}$	30	1	30
DEACTIVATE Pwd	14.2.3.4		$P_{deactivate_Pwd}$	30	1	30
DELETE Datei	14.2.4.1		P_{delete_EF}	100	1	100
DELETE Ordner			P_{delete_DF}	500	1	500
DELETE private Key	14.2.4.2		P_{delete_PrK}	60	1	60
DELETE sym. Key			P_{delete_SK}	60	1	60
DELETE PuK	14.2.4.3		P_{delete_PuK}	60	1	60
DELETE Pwd	14.2.4.4		P_{delete_Pwd}	60	1	60

Prüfpunkt				$T_{Ri} / [ms]$	g_i	$g_i T_{Ri} / [ms]$
LOAD APPLICATION	14.2.5		$P_{LoadApp}$	200	1	200
SELECT Datei	14.2.6.13	B.8.2	P_{select_EF}	10	800	8.000
SELECT Ordner	14.2.6.9	B.8.17	P_{select_DF}	10	350	3.500
TERMINATE CARD USAGE	14.2.7.1	B.8.4	$P_{terminateCard}$	30	1	30
TERMINATE DF	14.2.8.1	B.8.1	$P_{terminate_DF}$	30	1	30
TERMINATE EF	14.2.9.1		$P_{terminate_EF}$	30	1	30
TERMINATE privateKey	14.2.9.2		$P_{terminate_PrK}$	30	1	30
TERMINATE sym. Key			$P_{terminate_SK}$	30	1	30
TERMINATE PuK	14.2.9.3		$P_{terminate_PuK}$	30	1	30
TERMINATE Pwd	14.2.9.4		$P_{terminate_Pwd}$	30	1	30
SET LOGICAL EOF	14.3.4.1	B.8.5	P_{SetEOF}	600	1	600
ERASE BINARY wipe	14.3.1.1	B.8.6	$P_{WipeBin}$	600	1	600
READ BINARY b	14.3.2.1	B.8.7	$P_{ReadBinary,b}$	18	400	7.200
READ BINARY m			$P_{ReadBinary,m}$	60	400	24.000
UPDATE BINARY b	14.3.5.1	B.8.6	$P_{UpdateBin,b}$	50	200	10.000
UPDATE BINARY m			$P_{UpdateBin,m}$	70	200	14.000
WRITE BINARY b	14.3.6.1	B.8.5	$P_{wirte,b}$	50	1	50
WRITE BINARY m			$P_{wirte,m}$	70	1	70
ACTIVATE RECORD	14.4.1.1	B.8.8	$P_{ActivateRec}$	30	1	30
APPEND RECORD b	14.4.2.1		$P_{AppendRecord,b}$	70	200	14.000
APPEND RECORD m			$P_{AppendRecord,m}$	50	200	10.000
DEACTIVATE RECORD	14.4.3.1		$P_{DeactivateRec}$	30	1	30
DELETE RECORD	14.4.4.1		$P_{DeleteRec}$	80	1	80
ERASE RECORD wipe b	14.4.5.1		$P_{WipeRecord,b}$	40	1	40
ERASE RECORD wipe m			$P_{WipeRecord,m}$	80	1	80
READ RECORD b	14.4.6.1		$P_{ReadRecord,b}$	18	400	7.200
READ RECORD m			$P_{ReadRecord,m}$	60	400	24.000
SEARCH RECORD	14.4.7.1		$P_{SearchRec}$	120	1	120
UPDATE RECORD b	14.4.8.1		$P_{UpdateRecord,b}$	70	100	7.000
UPDATE RECORD m			$P_{UpdateRecord,m}$	50	100	5.000
CHANGE REFERENCE DATA	14.6.1.1	B.8.1	$P_{ChRefData}$	90	1	90
	14.6.1.2		P_{SetPIN}	50	1	50
DISABLE Ver. Req.	14.6.2.1		$P_{DisablePIN}$	50	1	50
ENABLE Ver. Req.	14.6.3.1		$P_{EnablePIN}$	50	1	50
GET PIN STATUS	14.6.4.1		$P_{GetPinStatus}$	10	1	10
RESET RETRY COUNTER	14.6.5.1		$P_{ResetRC}$	70	5	350
VERIFY	14.6.6.1		P_{Verify}	15	5	75
EXTERNAL AUTHENTICATE	14.7.1.1	B.8.11.4		0	0	0
		B.8.11.1	$P_{RoleCheck,ELC256}$	150	300	45.000
		B.8.11.2	$P_{RoleCheck,ELC384}$	200	100	20.000
		B.8.11.3	$P_{RoleCheck,ELC512}$	300	50	15.000
GET SECURITY STATUS KEY	14.7.3.1	B.8.10		0	0	0
	14.7.3.2	B.8.11.4		0	0	0
	14.7.3.3	B.8.11.1	$P_{SesKey,ELC256}$	10	3	30
INTERNAL AUTHENTICATE	14.7.4.1	B.8.12	$P_{RoleAuth,ELC256}$	120	300	36.000
			$P_{RoleAuth,ELC384}$	190	100	19.000
			$P_{RoleAuth,ELC512}$	280	50	14.000
				0	0	0
Sessionkeyaushandlung für Secure Messaging	15.4.1	B.8.10	$P_{SK4SM,AES128}$	140	200	28.000
			$P_{SK4SM,AES192}$	160	50	8.000
			$P_{SK4SM,AES256}$	180	10	1.800
	15.4.4	B.8.11.1		0	0	0
			$P_{SesKey,ELC256}$	400	210	84.000
			$P_{SesKey,ELC384}$	600	30	18.000
PSO Compute Digital Signature	14.8.2.1	B.8.13	$P_{signPSS,2048}$	270	100	27.000
			$P_{signPSS,3072}$	1.000	25	25.000
		B.8.14	$P_{signECDSA,256}$	100	100	10.000
			$P_{signECDSA,384}$	180	25	4.500
			$P_{signECDSA,512}$	280	10	2.800
PSO Decipher	14.8.3.1	B.8.15	$P_{dec,2048}$	250	60	15.000

Prüfpunkt				$T_{Ri} / [ms]$	g_i	$g_i T_{Ri} / [ms]$
	14.8.3.2	B.8.16	$P_{dec,3072}$	1.100	30	33.000
			$P_{dec,256}$	160	200	32.000
			$P_{dec,384}$	230	50	11.500
			$P_{dec,512}$	260	25	6.500
PSO Encipher	14.8.4.1	B.8.15	$P_{enc,2048}$	50	2,5	125
	14.8.4.2	B.8.16	$P_{enc,256}$	270	200	54.000
			$P_{enc,384}$	430	50	21.500
			$P_{enc,512}$	650	25	16.250
PSO Verify Certificate	14.8.7.2	B.8.11.1		0	0	0
			$P_{Import,ELC256}$	540	400	216.000
			$P_{Import,ELC384}$	680	200	136.000
			$P_{Import,ELC512}$	1.000	51	51.000
PSO Verify Digital Signature	14.8.9.1	B.8.14	$P_{verifyECDSA,256}$	100	5	500
			$P_{verifyECDSA,384}$	170	1	170
			$P_{verifyECDSA,512}$	270	1	270
FINGERPRINT	14.9.2.1	B.8.3	$P_{fingerprint}$	18.000	0,4	7.200
GENERATE ASYMMETRIC KEY PAIR	14.9.3.8	B.8.15	$P_{GAKP,2048}$	10.000	0,09	900
			$P_{GAKP,3072}$	60.000	0,01	600
		B.8.16	$P_{GAKP,256}$	180	100	18.000
			$P_{GAKP,384}$	270	50	13.500
GET CHALLENGE	14.9.4.2	B.8.17	$P_{challenge}$	10	200	2.000
MANAGE CHANNEL rst Ch	14.9.8.3		P_{reset_Ch}	5	10	50
MSE Restore	14.9.9.1	B.8.18	$P_{MSE_Restore}$	5	40	200
MSE Set	14.9.9		P_{MSE_Set}	10	300	3.000
Spaltensummen				104.738	10.000	1.220.380

Tabelle 275: Gesamtbewertung für Option_kontaktlose_Schnittstelle

Prüfpunkt				$T_{Ri} / [ms]$	g_i	$g_i T_{Ri} / [ms]$
GENERAL AUTHENTICATE	14.7.2.1	B.8.19	P_{PACE}	1.000	10	10.000
Spaltensummen				1.000	10	10.000

Tabelle 276: Gesamtbewertung für Option_Kryptobox

Prüfpunkt				$T_{Ri} / [ms]$	g_i	$g_i T_{Ri} / [ms]$
Sessionkeyaushandlung für Trusted Channel	15.4.1	B.8.20	$P_{SK4TC,AES128}$	90	15	1.350
			$P_{SK4TC,AES192}$	110	10	1.100
			$P_{SK4TC,AES256}$	120	15	1.800
PSO Compute Cryptographic Checksum	14.8.1.1	B.8.21	$P_{compute128,b}$	5	200	1.000
			$P_{compute128,m}$	10	200	2.000
			$P_{compute192,b}$	5	100	500
			$P_{compute192,m}$	15	100	1.500
			$P_{compute256,b}$	5	70	350
			$P_{compute256,m}$	20	70	1.400
PSO Decipher	14.8.3.3		$P_{dec128,b}$	5	200	1.000
			$P_{dec128,m}$	10	200	2.000
			$P_{dec192,b}$	5	100	500
			$P_{dec192,m}$	15	100	1.500
			$P_{dec256,b}$	5	70	350
			$P_{dec256,m}$	20	70	1.400
PSO Encipher	14.8.4.5		$P_{enc128,b}$	5	200	1.000
			$P_{enc128,m}$	10	200	2.000
			$P_{enc192,b}$	5	100	500
			$P_{enc192,m}$	15	100	1.500
			$P_{enc256,b}$	5	70	350
			$P_{enc256,m}$	20	70	1.400
PSO Verify Cryptographic Checksum	14.8.8.1		$P_{verify128,b}$	5	200	1.000
			$P_{verify128,m}$	10	200	2.000
			$P_{verify192,b}$	5	100	500

Prüfpunkt			$T_{Ri} / [ms]$	g_i	$g_i T_{Ri} / [ms]$
			$P_{verify192,m}$	15	100
			$P_{verify256,b}$	5	70
			$P_{verify256,m}$	20	70
Spaltensummen				560	3.000
					31.250

Tabelle 277: Gesamtbewertung für Option_logische_Kanäle

Prüfpunkt			$T_{Ri} / [ms]$	g_i	$g_i T_{Ri} / [ms]$
GET RANDOM	14.9.5.1	B.8.22	$P_{Random,b}$	10	10
			$P_{Random,m}$	1.000	1
MANAGE CHANNEL Open	14.9.8.1		P_{Open}	10	500
MANAGE CHANNEL Close	14.9.8.2	B.8.23	P_{Close}	5	400
MANAGE CHANNEL rst ICC	14.9.8.4		$PRST$	5	89
Spaltensummen				1.030	1.000
					8.545

Tabelle 278: Gesamtbewertung je nach Kombination der Optionen

Basis	Dual	Krypto	Kanal	Summe / [s]	Zulassungsgrenze / [s]
1.220.380	10.000	31.250	8.545	1270,2	555,7
1.220.380	10.000	31.250	0	1261,6	552,0
1.220.380	10.000	0	8.545	1238,9	542,0
1.220.380	10.000	0	0	1230,4	538,3
1.220.380	0	31.250	8.545	1260,2	551,3
1.220.380	0	31.250	0	1251,6	547,6
1.220.380	0	0	8.545	1228,9	537,6
1.220.380	0	0	0	1220,4	533,9

Hinweis (171): Bedeutung der Spaltenüberschriften in Tabelle 278:

- Basis: COS, welches keines der optionalen Funktionspakete enthält*
- Dual: COS enthält Option_kontaktlose_Schnittstelle*
- Krypto: COS enthält Option_Kryptobox*
- Kanal: COS enthält Option_logische_Kanäle*

(N109.500) K_externeWelt {K_Karte}

Für die Gesamtbewertung gilt:

$$P_{gesamt} = \sum_i g_i \cdot P_i = \sum_i g_i \cdot f_{1i} \cdot f_{2i} \cdot T_{Ri},$$

wobei nur die Prüfpunkte zu berücksichtigen sind, die gemäß der vom COS unterstützten Optionen möglich sind.

Unter den vereinfachenden Annahmen, dass die Messwerte innerhalb einer Reihe identisch sind (Standardabweichung σ ist null) und das Verhältnis X_i / T_{Ri} in allen Messreihen konstant ist, folgt dann für das Basisbetriebssystem:

$$P_{gesamt_einfach} = f_2 \sum_i g_i \cdot T_{Ri} = 1220,4s \cdot f_2, \text{ mit } f_2 \text{ als Funktion von } X_i / T_{Ri}.$$

Abbildung 12 zeigt diesen Zusammenhang graphisch mit dem Verhältnis X_i / T_{Ri} als Abszisse und $P_{gesamt_einfach}$ als Ordinate. Demnach erreicht ein unendlich schnelles Basisbetriebssystem 1220,4s Punkte. Eine Karte, die in allen Prüfpunkten die Referenzzeit benötigt, erreicht 915,3s Punkte. Eine Karte, die überall die doppelte Referenzzeit benötigt, null Punkte.

(N109.600) K_externeWelt {K_Karte}

Ein COS DARF KEINE Zulassung erhalten, wenn in wenigstens einem Prüfpunkt das Verhältnis X_i / T_{Ri} größer als vier ist.

(N109.700) K_externWelt {K_Karte}

Ein COS DARF KEINE Zulassung erhalten, wenn die gemäß (N109.500) ermittelte Punktzahl kleiner ist als in Tabelle 278 für die Kombination von Optionen angegeben. Dies entspricht einer relativen Ausführungszeit, die 1,5-mal so groß ist wie die Referenzzeit.

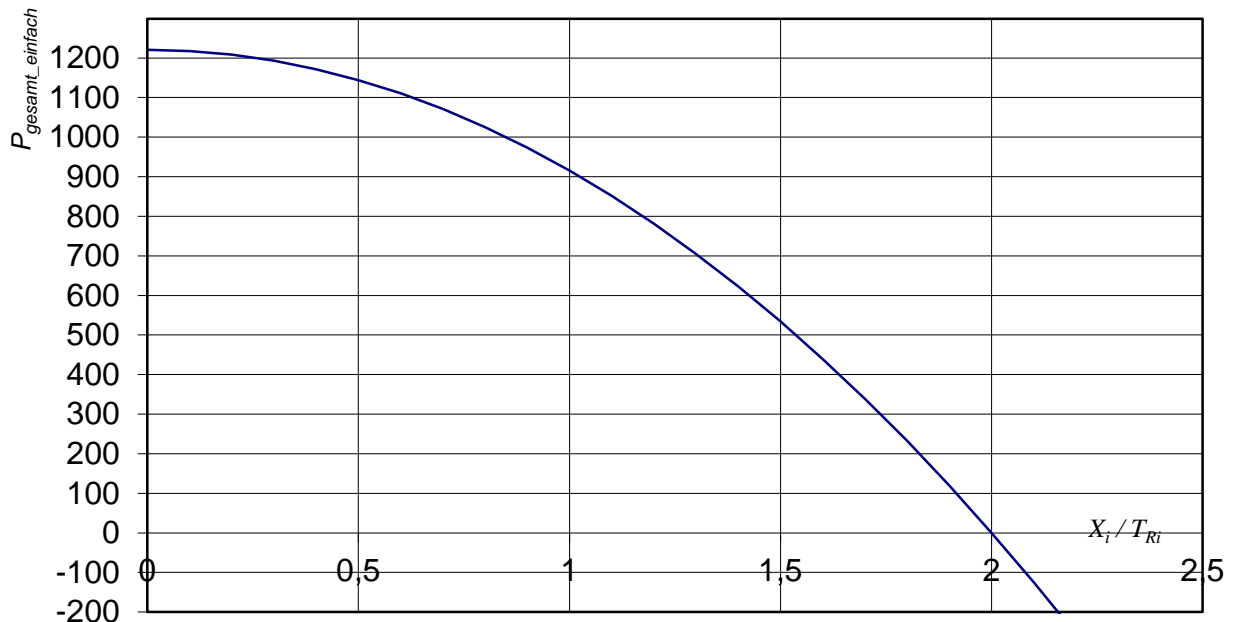


Abbildung 12: Graphische Darstellung von $P_{gesamt_einfach}$

B.6 Übertragungsgeschwindigkeit

B.6.1 Übertragungsgeschwindigkeit für das Übertragungsprotokoll T=1

Dieser Prüfpunkt berücksichtigt die Kanalkapazität des Kanals TPDU_Channel für das Übertragungsprotokoll T=1 in Abbildung 11. 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 Tabelle 273 ermittelt. Für die (rechnerische) Übertragungszeit von 1.000 Oktett zur Karte gilt:

$$t_T = \frac{1000 \cdot CGT}{C}$$

Mit C aus Tabelle 273 für den Fall $PPS1 = TA_1$ und CGT in Abhängigkeit von TC_1 im ATR gemäß folgender Tabelle:

Tabelle 279: 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

Es gilt: $P_{IO} = \text{points}(t_T, t_T, T_{IO})$.

B.6.2 Übertragungsgeschwindigkeit für das Protokoll [ISO/IEC 7816-12]

Hinweis (172): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

B.6.3 Übertragungsgeschwindigkeit für kontaktlose Datenübertragung

Hinweis (173): Die Anforderungen werden in einer späteren Dokumentenversion ergänzt.

Die folgenden Anforderungen 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)

B.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:

Keine.

Testdurchführung:

(N200.200) K_externeWelt {K_Karte}

Die Testdurchführung durchläuft eine Schleife 100-mal. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt. Abweichend von den Festlegungen in B.5.1 wird hier die Zeit pro Schleifendurchlauf anders ermittelt.

(N200.210) K_externeWelt {K_Karte}

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.

(N200.220) K_externeWelt {K_Karte}

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 TA₁ aus dem ATR verwendet werden.

(N200.230) K_externeWelt {K_Karte}

Schritt 3: Das IFD MUSS der Smartcard den Wert von IFSD = 254 präsentieren.

(N200.240) K_externeWelt {K_Karte}

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:

Es gilt: $P_{KarteStarten} = \text{points}((t_{Run4,1}, t_{Run4,2}, \dots, t_{Run4,100}), T_{KarteStarten})$.

Testnachbereitung:

Keine.

Besser wäre es, wenn auch hier die Messung aus B.5.1 anwendbar wäre. Es ist zu prüfen, ob dafür passende Equipment zur Verfügung steht.

B.8 Messverfahren für Einzelkommandos (normativ)

B.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:

(N201.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N201.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.LCS gesetzt werden.

Testdurchführung:

(N201.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 10 ausgeführt.

(N201.210) K_externeWelt {K_Karte}

Schritt 1: Die Datei / MF / DF.LCS / EF.LCS MUSS mittels Use Case aus 14.2.6.13

selektiert werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N201.220) K_externeWelt {K_Karte}

Schritt 2: LCS für Dateien:

- a. *currentEF* MUSS mittels Use Case aus 14.2.3.1 deaktiviert werden. Die Laufzeit $t_{deactivate_EF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- b. *currentEF* MUSS mittels Use Case aus 14.2.1.1 aktiviert werden. Die Laufzeit $t_{activate_EF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- c. *currentEF* MUSS mittels Use Case aus 14.2.9.1 terminiert werden. Die Laufzeit $t_{terminate_EF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- d. *currentEF* MUSS mittels Use Case aus 14.2.4.1 gelöscht werden. Die Laufzeit $t_{delete_EF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N201.230) K_externeWelt {K_Karte}

Schritt 3: Arbeiten mit Passwortobjekten:

- a. Der Status von PIN.LCS MUSS mittels Use Case aus 14.6.4.1 abgefragt werden. Die Laufzeit $t_{getStatus,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- b. Der Transportschutz von PIN.LCS MUSS mittels Use Case aus 14.6.1.2 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 B.5.1 gemessen werden.
- c. Das Attribut PIN.LCS.*flagEnabled* MUSS mittels Use Case aus 14.6.2.1 auf den Wert False gesetzt werden. Die Laufzeit $t_{disablePIN,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- d. Das Attribut PIN.LCS.*flagEnabled* MUSS mittels Use Case aus 14.6.3.1 auf den Wert True gesetzt werden. Die Laufzeit $t_{enablePIN,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- e. Das Attribut PIN.LCS.*secret* MUSS mittels Use Case aus 14.6.1.1 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 B.5.1 gemessen werden.
- f. VERIFY Kommando gemäß 14.6.6.1 mit korrekten *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{verify_1,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- g. VERIFY Kommando gemäß 14.6.6.1 mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{verify_2,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- h. VERIFY Kommando gemäß 14.6.6.1 mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{verify_3,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

- i. VERIFY Kommando gemäß 14.6.6.1 mit korrekten *verificationData* für das Objekt PIN.LCS. Die Laufzeit $t_{Verify_4,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- j. VERIFY Kommando gemäß 14.6.6.1 mit falschen *verificationData* für das Objekt PIN.LCS. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.
- k. RESET RETRY COUNTER gemäß 14.6.5.1 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 B.5.1 gemessen werden.

(N201.240) K_externeWelt {K_Karte}

Schritt 4: LCS für Passwortobjekte:

- a. PIN.LCS MUSS mittels Use Case aus 14.2.3.4 deaktiviert werden. Die Laufzeit $t_{deactivate_Pwd,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- b. PIN.LCS MUSS mittels Use Case aus 14.2.1.4 aktiviert werden. Die Laufzeit $t_{activate_Pwd,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- c. PIN.LCS MUSS mittels Use Case aus 14.2.9.4 terminiert werden. Die Laufzeit $t_{terminate_Pwd,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- d. PIN.LCS MUSS mittels Use Case aus 14.2.4.4 gelöscht werden. Die Laufzeit $t_{delete_Pwd,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N201.250) K_externeWelt {K_Karte}

Schritt 5: LCS für private Schlüssel:

- a. PrK.LCS MUSS mittels Use Case aus 14.2.3.2 deaktiviert werden. Die Laufzeit $t_{deactivate_PrK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- b. PrK.LCS MUSS mittels Use Case aus 14.2.1.2 aktiviert werden. Die Laufzeit $t_{activate_PrK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- c. PrK.LCS MUSS mittels Use Case aus 14.2.9.2 terminiert werden. Die Laufzeit $t_{terminate_PrK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- d. PrK.LCS MUSS mittels Use Case aus 14.2.4.2 gelöscht werden. Die Laufzeit $t_{delete_PrK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N201.260) K_externeWelt {K_Karte}

Schritt 6: LCS für öffentliche Schlüssel:

- a. PuK.LCS MUSS mittels Use Case aus 14.2.3.3 deaktiviert werden. Die Laufzeit $t_{deactivate_PuK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- b. PuK.LCS MUSS mittels Use Case aus 14.2.1.3 aktiviert werden. Die Laufzeit $t_{activate_PuK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

- c. PuK.LCS MUSS mittels Use Case aus 14.2.9.3 terminiert werden. Die Laufzeit $t_{terminate_PuK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- d. PuK.LCS MUSS mittels Use Case aus 14.2.4.3 gelöscht werden. Die Laufzeit $t_{delete_PuK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N201.270) K_externeWelt {K_Karte}

Schritt 7: LCS für symmetrische Schlüssel:

- a. SK.LCS MUSS mittels Use Case aus 14.2.3.2 deaktiviert werden. Die Laufzeit $t_{deactivate_SyK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- b. SK.LCS MUSS mittels Use Case aus 14.2.1.2 aktiviert werden. Die Laufzeit $t_{activate_SyK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- c. SK.LCS MUSS mittels Use Case aus 14.2.9.2 terminiert werden. Die Laufzeit $t_{terminate_SyK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- d. SK.LCS MUSS mittels Use Case aus 14.2.4.2 gelöscht werden. Die Laufzeit $t_{delete_SyK,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N201.280) K_externeWelt {K_Karte}

Schritt 8: Der Ordner / MF / DF.LCS MUSS mittels Use Case aus 14.2.6.9 selektiert werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N201.290) K_externeWelt {K_Karte}

Schritt 9: LCS für Ordner:

- a. *currentFolder* MUSS mittels Use Case aus 14.2.3.1 deaktiviert werden. Die Laufzeit $t_{deactivate_DF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- b. *currentFolder* MUSS mittels Use Case aus 14.2.1.1 aktiviert werden. Die Laufzeit $t_{activate_DF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- c. *currentFolder* MUSS mittels Use Case aus 14.2.8.1 terminiert werden. Die Laufzeit $t_{terminate_DF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- d. *currentFolder* MUSS mittels Use Case aus 14.2.4.1 gelöscht werden. Die Laufzeit $t_{delete_DF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N201.300) K_externeWelt {K_Karte}

Schritt 10: Laden von DF.LCS.

- a. 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.
- b. Die Ausführungszeit jedes LOAD APPLICATION Kommandos MUSS gemäß B.5.1 gemessen werden.

- c. Die Summe aller LOAD APPLICATION Kommandoausführungszeiten in der i -ten Schleifeniteration wird mit $t_{LoadApp,i}$ bezeichnet.

Testauswertung:

Es gilt:

$P_{activate_DF}$	=	points($(t_{activate_DF,1},$	$t_{activate_DF,2}, \dots,$	$t_{activate_DF,100})$,	$T_{activate_DF}$
$P_{activate_EF}$	=	points($(t_{activate_EF,1},$	$t_{activate_EF,2}, \dots,$	$t_{activate_EF,100})$,	$T_{activate_EF}$
$P_{activate_Pwd}$	=	points($(t_{activate_Pwd,1},$	$t_{activate_Pwd,2}, \dots,$	$t_{activate_Pwd,100})$,	$T_{activate_Pwd}$
$P_{activate_PrK}$	=	points($(t_{activate_PrK,1},$	$t_{activate_PrK,2}, \dots,$	$t_{activate_PrK,100})$,	$T_{activate_PrK}$
$P_{activate_PuK}$	=	points($(t_{activate_PuK,1},$	$t_{activate_PuK,2}, \dots,$	$t_{activate_PuK,100})$,	$T_{activate_PuK}$
$P_{activate_SK}$	=	points($(t_{activate_SK,1},$	$t_{activate_SK,2}, \dots,$	$t_{activate_SK,100})$,	$T_{activate_SK}$
$P_{deactivate_DF}$	=	points($(t_{deactivate_DF,1},$	$t_{deactivate_DF,2}, \dots,$	$t_{deactivate_DF,100})$,	$T_{deactivate_DF}$
$P_{deactivate_EF}$	=	points($(t_{deactivate_EF,1},$	$t_{deactivate_EF,2}, \dots,$	$t_{deactivate_EF,100})$,	$T_{deactivate_EF}$
$P_{deactivate_Pwd}$	=	points($(t_{deactivate_Pwd,1},$	$t_{deactivate_Pwd,2}, \dots,$	$t_{deactivate_Pwd,100})$,	$T_{deactivate_Pwd}$
$P_{deactivate_PrK}$	=	points($(t_{deactivate_PrK,1},$	$t_{deactivate_PrK,2}, \dots,$	$t_{deactivate_PrK,100})$,	$T_{deactivate_PrK}$
$P_{deactivate_PuK}$	=	points($(t_{deactivate_PuK,1},$	$t_{deactivate_PuK,2}, \dots,$	$t_{deactivate_PuK,100})$,	$T_{deactivate_PuK}$
$P_{deactivate_SK}$	=	points($(t_{deactivate_SK,1},$	$t_{deactivate_SK,2}, \dots,$	$t_{deactivate_SK,100})$,	$T_{deactivate_SK}$
P_{delete_DF}	=	points($(t_{delete_DF,1},$	$t_{delete_DF,2}, \dots,$	$t_{delete_DF,100})$,	T_{delete_DF}
P_{delete_EF}	=	points($(t_{delete_EF,1},$	$t_{delete_EF,2}, \dots,$	$t_{delete_EF,100})$,	T_{delete_EF}
P_{delete_Pwd}	=	points($(t_{delete_Pwd,1},$	$t_{delete_Pwd,2}, \dots,$	$t_{delete_Pwd,100})$,	T_{delete_Pwd}
P_{delete_PrK}	=	points($(t_{delete_PrK,1},$	$t_{delete_PrK,2}, \dots,$	$t_{delete_PrK,100})$,	T_{delete_PrK}
P_{delete_PuK}	=	points($(t_{delete_PuK,1},$	$t_{delete_PuK,2}, \dots,$	$t_{delete_PuK,100})$,	T_{delete_PuK}
P_{delete_SK}	=	points($(t_{delete_SK,1},$	$t_{delete_SK,2}, \dots,$	$t_{delete_SK,100})$,	T_{delete_SK}
$P_{terminate_DF}$	=	points($(t_{terminate_DF,1},$	$t_{terminate_DF,2}, \dots,$	$t_{terminate_DF,100})$,	$T_{terminate_DF}$
$P_{terminate_EF}$	=	points($(t_{terminate_EF,1},$	$t_{terminate_EF,2}, \dots,$	$t_{terminate_EF,100})$,	$T_{terminate_EF}$
$P_{terminate_Pwd}$	=	points($(t_{terminate_Pwd,1},$	$t_{terminate_Pwd,2}, \dots,$	$t_{terminate_Pwd,100})$,	$T_{terminate_Pwd}$
$P_{terminate_PrK}$	=	points($(t_{terminate_PrK,1},$	$t_{terminate_PrK,2}, \dots,$	$t_{terminate_PrK,100})$,	$T_{terminate_PrK}$
$P_{terminate_PuK}$	=	points($(t_{terminate_PuK,1},$	$t_{terminate_PuK,2}, \dots,$	$t_{terminate_PuK,100})$,	$T_{terminate_PuK}$
$P_{terminate_SK}$	=	points($(t_{terminate_SK,1},$	$t_{terminate_SK,2}, \dots,$	$t_{terminate_SK,100})$,	$T_{terminate_SK}$

$P_{LoadApp}$	=	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($t_{Verify_1,1}$,	$t_{Verify_1,2}$, ...,	$t_{Verify_1,100}$),	T_{Verify_1}
P_{Verify_2}	=	points($t_{Verify_2,1}$,	$t_{Verify_2,2}$, ...,	$t_{Verify_2,100}$),	T_{Verify_2}
P_{Verify_3}	=	points($t_{Verify_3,1}$,	$t_{Verify_3,2}$, ...,	$t_{Verify_3,100}$),	T_{Verify_3}
P_{Verify_4}	=	points($t_{Verify_4,1}$,	$t_{Verify_4,2}$, ...,	$t_{Verify_4,100}$),	T_{Verify_4}
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.

B.8.2 SELECT Datei

In diesem Kapitel wird die Selektion von Dateien betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.SelectEF.

Testvorbereitung:

(N202.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N202.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.SelectEF gesetzt werden.

(N202.130) K_externeWelt {K_Karte}

Schritt 3: *currentEF* MUSS auf / MF / DF.SelectEF / EF.00 gesetzt werden.

Testdurchführung:

(N202.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.

(N202.210) K_externeWelt {K_Karte}

Schritt 1: Aus der Menge der zu Anfang 100 in DF.SelectEF gültigen Fileidentifizierer wird ein bislang noch nicht verwendeter gezogen (ziehen ohne zurücklegen).

(N202.220) K_externeWelt {K_Karte}

Schritt 2: Die Datei mit dem in Schritt 1 ausgewählten Fileidentifizierer MUSS mittels

Use Case aus 14.2.6.13 selektiert werden. Die Laufzeit $t_{select_EF,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{select_EF} = \text{points}((t_{select_EF,1}, t_{select_EF,2}, \dots, t_{select_EF,100}), T_{select_EF})$$

Testnachbereitung:

Keine.

B.8.3 FINGERPRINT

In diesem Abschnitt wird die Berechnung des COS Fingerprints betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF.

Testvorbereitung:

(N203.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

(N203.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.

(N203.210) K_externeWelt {K_Karte}

Schritt 1: In der i -ten Schleifeniteration MUSS ein Oktettstring $prefix = \text{RAND}(128)$ erzeugt werden.

(N203.220) K_externeWelt {K_Karte}

Schritt 2: FINGERPRINT Kommando gemäß 14.9.2.1, wobei als $prefix$ der im vorherigen Schritt erzeugte Wert dient. Die Laufzeit $t_{fp,i}$ dieses Kommandos MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{fingerprint} = \text{points}((t_{fp,1}, t_{fp,2}, \dots, t_{fp,100}), T_{fingerprint})$$

Testnachbereitung:

Keine.

B.8.4 TERMINATE CARD USAGE

In diesem Kapitel wird das Terminieren einer Smartcard betrachtet.

Testvorbereitung:

(N204.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

(N204.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife zehnmal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.

(N204.210) K_externeWelt {K_Karte}

Schritt 1: Die Smartcard MUSS mittels Use Case aus 14.2.7.1 terminiert werden. Die Laufzeit $t_{terminateCard,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N204.220) K_externeWelt {K_Karte}

Schritt 2: Die Smartcard MUSS so reinitialisiert werden, dass die in B.9 definierte Kartenkonfiguration vorliegt.

Testauswertung:

Es gilt:

$$P_{terminateCard} = \text{points}((t_{terminateCard,1}, t_{terminateCard,2}, \dots, t_{terminateCard,10}), T_{terminateCard})$$

Testnachbereitung:

Keine.

B.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:

(N205.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N205.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.

(N205.130) K_externeWelt {K_Karte}

Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.

(N205.140) K_externeWelt {K_Karte}

Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen $M_{Nc} = \{1, 2, 3, \dots, 1000\}$ erstellt werden.

(N205.150) K_externeWelt {K_Karte}

Schritt 5: Es MUSS eine leere Menge $M_{SetEOF} = \{\}$ erstellt werden.

Testdurchführung:

(N205.200) K_externeWelt {K_Karte}

Die Testdurchführung durchläuft 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.

(N205.210) K_externeWelt {K_Karte}

Schritt 1: Aus der Menge M_{Nc} wird ein beliebiges Element e zufällig gezogen (ziehen ohne zurücklegen) und der Oktettstring $newData = \text{RAND}(e)$ erzeugt.

(N205.220) K_externeWelt {K_Karte}

Schritt 2: Der *body* von EF.transparent wird gemäß Use Case aus 14.3.6.1 erweitert wobei *newData* als Kommandonachricht verwendet wird. Die Laufzeit $t_{write,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N205.230) K_externeWelt {K_Karte}

Schritt 3: Falls das Attribut EF.transparent.*positionLogicalEndOfFile* nach Schritt 2 größer als '7800' = 30.720 ist, dann MUSS dieses Attribut mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt werden. Die Laufzeit t_{set} dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge M_{SetEOF} hinzugefügt werden.

(N205.240) K_externeWelt {K_Karte}

Schritt 4: Das Attribut EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt. Die Laufzeit t_{set} dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge M_{SetEOF} hinzugefügt werden.

Testauswertung:

Durch die Messpunkt $(x, y) \rightarrow (e, t_{write,i})$ wird eine Ausgleichsgerade $y = m x + b$ gelegt.

Es gilt:

$$P_{write,b} = \text{points}((b, b) , T_{write,b})$$

$$P_{write,m} = \text{points}((1000m, T_{write,b}), (1000m, 1000m))$$

$$P_{SetEOF} = \text{points}(M_{SetEOF} , T_{SetEOF})$$

Hinweis (174): 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.

B.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:

(N206.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N206.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.

(N206.130) K_externeWelt {K_Karte}

Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.

(N206.140) K_externeWelt {K_Karte}

Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen $M_{Nc} = \{1, 2, 3, \dots, 1000\}$ erstellt werden.

(N206.150) K_externeWelt {K_Karte}

Schritt 5: Es MUSS eine leere Menge $M_{wipe} = \{\}$ erstellt werden.

(N206.160) K_externeWelt {K_Karte}

Schritt 6: Das Attribut EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus 14.3.5.1 auf den Wert EF.transparent.*numberOfOctet* gesetzt, wobei für den Kommandoparameter *newData* = '00' gilt.

(N206.170) K_externeWelt {K_Karte}

Schritt 7: Die Variable *index* wird auf den Wert null gesetzt.

Testdurchführung:

(N206.200) K_externeWelt {K_Karte}

Die Testdurchführung durchläuft 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.

(N206.210) K_externeWelt {K_Karte}

Schritt 1: Aus der Menge M_{Nc} wird ein beliebiges Element *e* zufällig gezogen (ziehen ohne zurücklegen) und der Oktettstring *newData* = RAND(*e*) erzeugt.

(N206.220) K_externeWelt {K_Karte}

Schritt 2: Der *body* von EF.transparent wird gemäß Use Case aus 14.3.5.1 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 B.5.1 gemessen werden. Nach Kommandoausführung wird *index* += *e* gesetzt.

(N206.230) K_externeWelt {K_Karte}

Schritt 3: Falls *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 14.3.1.1 gelöscht werden, wobei der Kommandoparameter *offset* = 0 gesetzt wird. Die Laufzeit t_{wipe} dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge M_{wipe} hinzugefügt werden.

(N206.240) K_externeWelt {K_Karte}

Schritt 4: Der Inhalt von EF.transparent wird mittels Use Case aus 14.3.1.1 gelöscht, wobei der Kommandoparameter *offset* = 0 gesetzt wird. Die Laufzeit t_{wipe} dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge M_{wipe} hinzugefügt werden.

Testauswertung:

Durch die Messpunkt $(x, y) \rightarrow (e, t_{update,i})$ wird eine Ausgleichsgerade $y = m x + b$ gelegt.

Es gilt:

$$P_{UpdateBin,b} = \text{points}((b, b), T_{update,b})$$

$$P_{UpdateBin,m} = \text{points}((1000m, 1000m), T_{update,m})$$

$$P_{WipeBin} = \text{points}(M_{wipe}, T_{wipe})$$

Hinweis (175): 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:

(N206.300) K_externeWelt {K_Karte}

Das Attribut EF.transparent.*positionLogicalEndOfFile* wird mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt.

B.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:

(N207.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N207.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.transparent gesetzt werden.

(N207.130) K_externeWelt {K_Karte}

Schritt 3: *currentEF* MUSS auf / MF / DF.transparent / EF.transparent gesetzt werden.

(N207.140) K_externeWelt {K_Karte}

Schritt 4: Es MUSS eine Menge mit den natürlichen Zahlen $M_{Ne} = \{1, 2, 3, \dots, 1000\}$ erstellt werden.

(N207.150) K_externeWelt {K_Karte}

Schritt 5: Das Attribut EF.transparent.*body* MUSS vollständig mit zufälligen Werten befüllt werden.

Testdurchführung:

(N207.200) K_externeWelt {K_Karte}

Die Testdurchführung durchläuft eine Schleife solange, bis die Menge M_{Ne} leer ist. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.

(N207.210) K_externeWelt {K_Karte}

Schritt 1: Aus der Menge M_{Ne} wird ein beliebiges Element e zufällig gezogen (ziehen ohne zurücklegen).

(N207.220) K_externeWelt {K_Karte}

Schritt 2: Teile von EF.transparent.*body* werden gemäß Use Case aus 14.3.2.1 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 B.5.1 gemessen werden.

Testauswertung:

Durch die Messpunkt $(x, y) \rightarrow (e, t_{read,i})$ wird eine Ausgleichsgerade $y = m x + b$ gelegt.

Es gilt:

$$P_{ReadBinary,b} = \text{points}((b, b) , T_{ReadBinary,b})$$

$$P_{ReadBinary,m} = \text{points}((1000m, 1000m) , T_{ReadBinary,m})$$

Hinweis (176): 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:

(N207.300) K_externeWelt {K_Karte}

Das Attribut EF.transparent.positionLogicalEndOfFile wird mittels Use Case aus 14.3.4.1 auf den Wert null gesetzt.

B.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 B.8.9 behandelt. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.strukturiert / EF.strukturiert.

Testvorbereitung:

(N208.100) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N208.110) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.strukturiert gesetzt werden.

(N208.120) K_externeWelt {K_Karte}

Schritt 3: Die Datei / MF / DF.strukturiert / EF.strukturiert MUSS mittels Use Case aus 14.2.6.13 selektiert werden.

Testdurchführung:

(N208.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 255 mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 7 ausgeführt.

(N208.210) K_externeWelt {K_Karte}

Schritt 1, APPEND RECORD: Im i -ten Schleifendurchlauf wird ein Oktettstring *recordData* = RAND(i) erzeugt. In *currentEF* MUSS mittels Use Case aus 14.4.2.1 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 B.5.1 gemessen werden.

(N208.220) K_externeWelt {K_Karte}

Schritt 2, DEACTIVATE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.3.1 deaktiviert werden. Die Laufzeit $t_{deactivate,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N208.230) K_externeWelt {K_Karte}

Schritt 3, ACTIVATE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.1.1 aktiviert werden. Die Laufzeit $t_{activate,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N208.240) K_externeWelt {K_Karte}

Schritt 4, UPDATE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.8.1 auf den Wert *newData* = RAND(i) werden. Die Laufzeit $t_{update,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N208.250) K_externeWelt {K_Karte}

Schritt 5, READ RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus

14.4.6.1 gelesen werden. Die Laufzeit $t_{read,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N208.260) K_externeWelt {K_Karte}

Schritt 6, ERASE RECORD wipe: Der Inhalt von Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.5.1 gelöscht werden. Die Laufzeit $t_{wipe,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N208.270) K_externeWelt {K_Karte}

Schritt 7, DELETE RECORD: Rekord 1 in *currentEF* MUSS mittels Use Case aus 14.4.4.1 aus *currentEF.recordList* entfernt werden. Die Laufzeit $t_{delete,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Durch die Messpunkt $(x, y) \rightarrow (i, t_{a,i})$ wird eine Ausgleichsgerade $y = m_a x + b_a$ gelegt.

Es gilt:

$$P_{AppendRecord,b} = \text{points}((b_{AppendRecord}, b_{AppendRecord}), T_{AppendRecord,b})$$

$$P_{AppendRecord,m} = \text{points}((1000m_{AppendRecord}, 1000m_{AppendRecord}), T_{AppendRecord,m})$$

$$P_{UpdateRecord,b} = \text{points}((b_{UpdateRecord}, b_{UpdateRecord}), T_{UpdateRecord,b})$$

$$P_{UpdateRecord,m} = \text{points}((1000m_{UpdateRecord}, 1000m_{UpdateRecord}), T_{UpdateRecord,m})$$

$$P_{ReadRecord,b} = \text{points}((b_{ReadRecord}, b_{ReadRecord}), T_{ReadRecord,b})$$

$$P_{ReadRecord,m} = \text{points}((1000m_{ReadRecord}, 1000m_{ReadRecord}), T_{ReadRecord,m})$$

$$P_{WipeRecord,b} = \text{points}((b_{WipeRecord}, b_{WipeRecord}), T_{WipeRecord,b})$$

$$P_{WipeRecord,m} = \text{points}((1000m_{WipeRecord}, 1000m_{WipeRecord}), T_{WipeRecord,m})$$

Hinweis (177): 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.

$$P_{ActivateRec} = \text{points}((t_{ActivateRec,1}, t_{ActivateRec,2}, \dots, t_{ActivateRec,1000}), T_{ActivateRec})$$

$$P_{DeactivateRec} = \text{points}((t_{DeactivateRec,1}, t_{DeactivateRec,2}, \dots, t_{DeactivateRec,1000}), T_{DeactivateRec})$$

$$P_{DeleteRec} = \text{points}((t_{DeleteRec,1}, t_{DeleteRec,2}, \dots, t_{DeleteRec,1000}), T_{DeleteRec})$$

Testnachbereitung:

Keine.

B.8.9 SEARCH RECORD

Hier wird lediglich das Kommando SEARCH RECORD betrachtet. Die übrigen rekordorientierten Kommandos werden in B.8.8 behandelt. Dieser Prüfpunkt arbeitet mit der Datei / MF / DF.strukturiert / EF.strukturiert.

Testvorbereitung:

(N209.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N209.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.strukturiert gesetzt werden.

(N209.130) K_externeWelt {K_Karte}

Schritt 3: Die Datei / MF / DF.strukturiert / EF.strukturiert MUSS mittels Use Case aus 14.2.6.13 selektiert werden.

(N209.140) K_externeWelt {K_Karte}

Schritt 4: Das Attribut *currentEF.recordList* MUSS mittels Use Case aus 14.4.2.1 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 = 'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF1000FF...FF',
- f. Rekord 252 = 'FF...FFFC00FFFF',
- g. Rekord 253 = 'FF...FFFD00FF',
- h. Rekord 254 = 'FF...FFFE00'.

Hinweis (178): 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:

(N209.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 254-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt. Nach Abarbeitung der Schleife wird mit Schritt 3 fortgefahren.

(N209.210) K_externeWelt {K_Karte}

Schritt 1: Im *i*-ten Schleifendurchlauf gilt: *searchString* = I2OS(*i*, 1).

(N209.220) K_externeWelt {K_Karte}

Schritt 2: In *currentEF* MUSS mittels Use Case aus 14.4.7.1 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 B.5.1 gemessen werden.

(N209.230) K_externeWelt {K_Karte}

Schritt 3: In *currentEF* MUSS mittels Use Case aus 14.4.7.1 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 B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{SearchRec} = \text{points}((t_{SearchRec,1}, t_{SearchRec,2}, \dots, t_{SearchRec,255}), T_{SearchRec})$$

Testnachbereitung:

(N209.300) K_externeWelt {K_Karte}

Aus / MF / DF.strukturiert / EF.strukturiert MÜSSEN mittels Use Case aus 14.4.4.1 MÜSSEN alle Rekords entfernt werden.

B.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:

(N210.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N210.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

(N210.130) K_externeWelt {K_Karte}

Schritt 3: Es MUSS eine leere Menge $M_{GetStatus} = \{\}$ erstellt werden.

Testdurchführung:

(N210.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS 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.

(N210.210) K_externeWelt {K_Karte}

Schritt 1: MSE Set Kommando gemäß 14.9.9.6, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4SM gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N210.220) K_externeWelt {K_Karte}

Schritt 2: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 6 ausgeführt. Anschließend wird mit Schritt 7 fortgefahren.

(N210.230) K_externeWelt {K_Karte}

Schritt 3: Es wird eine Zufallszahl mittels Use Case aus 14.9.4.2 vom Prüfling abgeholt. Die Laufzeit $t_{rand,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N210.240) K_externeWelt {K_Karte}

Schritt 4: Es wird eine erfolgreiche gegenseitige Authentisierung gemäß Use Case aus 14.7.1.2 und (N084.410)a durchgeführt. Die Laufzeit $t_{Auth,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N210.250) K_externeWelt {K_Karte}

Schritt 5: Die Ausführungszeiten werden wie folgt zusammengefasst:

$$t_{SK,i} = t_{rand,i} + t_{Auth,i}$$

(N210.260) Dieser Punkt ist absichtlich leer.

(N210.270) K_externeWelt {K_Karte}

Schritt 7: MSE Restore Kommando gemäß 14.9.9.1 mit $seNo = 1$, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

Testauswertung:

Es gilt:

$$P_{SK4SM,AES128} = \text{points}(t_{SK128,1}, t_{SK128,2}, \dots, t_{SK128,100}), T_{SK4SM,AES128}$$

$$P_{SK4SM,AES192} = \text{points}(t_{SK192,1}, t_{SK192,2}, \dots, t_{SK192,100}), T_{SK4SM,AES192}$$

$$P_{SK4SM,AES256} = \text{points}(t_{SK256,1}, t_{SK256,2}, \dots, t_{SK256,100}), T_{SK4SM,AES256}$$

Testnachbereitung:

Keine.

B.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 der eGK) sowie die asymmetrische Aushandlung von Session-keys 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.

B.8.11.1 ELC 256

Es werden zehn Test CA benutzt, denen von der Root-CA (siehe PuK.RCA_ELC256) folgende CV-Zertifikate zugeordnet werden:

Test CA0	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d0403028641043dc8968bf65157711a3547714230e7d80667d1451897204a9fcd91e5e53b1aeb960a4257e36656f06da638d50d0a4095fbfa11c99c8481da49fe5245a47638855f200844455858586f01127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3740960632d2f613697ccbb05b3b21ae6ddb1fbb146deb5cc3f7787a28d485c2e7042b5001db598eac960055c56e3489c568302ff3e638a8b5bec070995859629a9 '
----------	--

Test CA1	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104068fb3f80118c9a1390b6f3d6d84a5dd5f5c1228e57268afd09297b71c5d8d451879b7097be706fb9a9e6952f599ef0f7eaa13f111d9c1ec656fdc93163ed9575f200844455858586f11127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37406fc05554871d4f0cc60386b500e21975471265ca08a47b932082b596e8be8d bc6e68a729051c47b38db23e16667ff4fae724e39529c7ebd8f05aab249e824c9b'
Test CA2	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d0403028641045fe0d65046d587cf0d494766ed0f4acd646f00820c7d98458df25f49aab891cd2b80e5d51c18719354bf8c8f3a83a41ec8819e894a55a27b8acfe212393f4e685f200844455858586f21127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37404f73ecc1733ce8acba3c539b45ba3ba79829a65d6d1057f3f3732fd32c1920 ca8efb4489072cc4943b4b948421b146053f39bd51c5275d534ec395333b9f6430'
Test CA3	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104a2f12f29f8c45da68c2d9d484ef4f764b861934f19c8ad829edad541050blad9323d830837144714bac3ff9ffa9b9f2ec5288e93373744ba2923c7afb72e6fde5f200844455858586f31127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f374048a2a0e899ac08817249726f1f13bb983ee5728d3439869953b46b8b69c41d02690ebelbe8a04a5897c82ca04eedc527a016d798f07a776d75fa99f2c7f88e7f'
Test CA4	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104156028235673196c4d54e4b78205c9979d2bdb39362d05568ee5c11671ac9010284bb19e24444b6f174167cd50efd5e6ef9a5780788ac02312f6961dd180e9785f200844455858586f41127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f374081d4965d98f35605fcb6bb6303378787810af9182d2b9fb495b22161a29e553a60ea87f7182c96153dc7ef1dfeae79e32ad5dae554ecalc244ab73f57d76cf14'
Test CA5	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d04030286410462082556bbab7c0387eb3ee7a2767a2277beaf9a02ccdc255a07baf97b232a74c18c584888ac8f655c05d56ac7afe399c79bfb865e6fe71ee8a76f35257ff0d5f200844455858586f51127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37408db6a883d4e575c1667edd7fc1b3elb41e7a5d8f73c8ce81b29232f671f4b5178ea62299d06eae35c09794233ad1979b2e263449dbf4c9d68d74331a9a3d94eb'
Test CA6	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d0403028641041932b906841a94133354ef5fb2878d29612e83e3f55b43aceb44d809bleebcd6c0c515dece9c483273b5f070c3ab2924a65e26900e73506ed46d96be09bfbfad5f200844455858586f61127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3740948249b88e2d1db43237cdc5c814865aed1ac1f8685cfcece3222c2bd169c23c15fe199585d714e74fd468893299f739f80c916e57dff8fffb1747624809122fa3'
Test CA7	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d040302864104827bc42213603730397a0d1a25981fbf3d4c4326ebf943784fba310eebd0af39153d457fb56d7dacfe7d3acc0397fa29160c67f0fc11caf4d9074b79213355bc5f200844455858586f71127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37402bca5198dceee074b1a1ed84dddb52c06ea11e190aaa938636844aaba27741207a66ac35471fa81fca9d04dea2e410fcf1757328fe9d8deec645e5707bdb8cad'
Test CA8	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d04030286410427ed1d352e39f0f029f07314414df5a2070587e316f42dee798cf117e9dbfbabb56a77fb65169d1014b950eeb12efdcdd6e63d44e351386457b53ec58375d10c65f200844455858586f81127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3740953e492d4fcd165cf0aa658aa0c3cccf4bf4877ec52acdb3eb1b2ba2810e8f30928563f2ce1e3666dac086ef84c749c29d6dfb21b448c67b435fe85bdfd126d'
Test CA9	'7f2181d87f4e81915f290170420844455858586001127f494d06082a8648ce3d04030286410437b7e999de3102cfab44162d204b1bcab1d495da3f3c1293db90198b2399f42920d2b716e8e5fceebea0a760df6a27009b4f5b5a51a4dae74c4e0aa03d9ab16515f200844455858586f91127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f374006c12b8bb2affd0e2dc64dc9a51df59ac7d0685b736c58a5af92c0e7c0147e151f937657c6500bb3361c9b504ec39f3867990201ff78d238aa6e797aa74f59c7'

Test:

(N211.110) K_externWelt {K_Karte}

Dieser Prüfpunkt arbeitet mit folgenden Parametern:

a. M_{CVC} = {TestCA0, TestCA1, ..., TestCA9}

b. $PuK.RCA$ = / MF / DF.Auth / PuK.RCA_ELC256

c. $PrK.Auth$ = / MF / DF.Auth / PrK.Auth_ELC256

(N211.120) K_externeWelt {K_Karte}

Die Performanzmessung wird wie folgt durchgeführt:

$(M_{Import}, M_{RoleCheck}, M_{SesKey}, M_{GetSecSta}) = PerformanceAuth(M_{CVC}, PuK.RCA, PrK.Auth)$

Testauswertung:

Es gilt:

$P_{Import, ELC256} = points(M_{Import}, T_{Import, ELC256})$

$P_{RoleCheck, ELC256} = points(M_{RoleCheck}, T_{RoleCheck, ELC256})$

$P_{SesKey, ELC256} = points(M_{SesKey}, T_{SesKey, ELC256})$

$P_{GetSecStat, ELC256} = points(M_{GetSecStat}, T_{GetSecStat, ELC256})$

B.8.11.2 ELC 384

Es werden zehn Test CA benutzt, denen von der Root-CA (siehe PuK.RCA_ELC384) folgende CV-Zertifikate zugeordnet werden:

Test CA0	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d04030386610450130d536c93d3183e4e7271fd291f34be2dde0869749b19420833220f10d4d8ae61f6567710c277fca109565dea64c184f6bdea3470c2082283fcf58ee436983d90f91e63252747282cbe4218c1befcfd8bf7178be5dc0d0dd9a558a999b63e5f200844455858586f02127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37605aed2ebd167b7872df46b37644afdac7d836602365025bdfb58af88404c3986e97b0771677aafc326bc46e2ae3ff6c72718ad232da064b7ff31aec2822d78dd8c4e56613e233a0e76fbf92ad2bf8d6704189a11910f5e94fb45436b578b0894c'
Test CA1	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d0403038661048c24bfff918480325f3556cf426460972c797e3ba9ea9c7clad0daccdd29d3a09fcc9a4fc30e7b0f23f99bb206c9a9a2722b9fa21bcd759368137edc309a31214d610cbe08ba9590215c646cbb54b3548268884afc5e4f84c7fa86c8fcb091f595f200844455858586f12127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f3760348a55a41fe1624641874b5b10f31667885374ac9db23e67cc75f86895ffa279e444ecb7c0361fabe13ba556056900524f309700775f9774de0fb78371c661902cc11f19bb059e8a32170afd06ce4e47c19ad6b1a59ef99b993159354d8d8f95'
Test CA2	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d0403038661047c07cbafca6624495a06ab1cc0bbc42210181cd0ba12eb687eb78dafd77a65ec8314aaa fcaadbada5c071ef705857c7644807136b106d07befc1be5279e7b4536bd7fc70016ce20b448479594441e8179ee4eeda328cc5034756b6871eede6a6c5f200844455858586f22127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f376035f2154ab00b3d4df8abbed23c261336a833fb977cc18f4dc570754bb89b91e62ffa1605609a0c687f64f88ea0bc83ad4c8c615cb76868b73f7e55bef0bb3e0db3043d386f1faf32e477a2f0803965bda9ac9a5afecf8d29ae6edfddb4bd2c3e'
Test CA3	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d0403038661046aec3041253a998dde1b7f060abeab85939b9c6ab6dd32d7f01b03736d22154c63753c375ae7783189484be4c94772bc5e39cf3b02bf35218c0b1f63bfc234485bcdea8bfe55c39e32928be7ca61638321b03b5dd1912960079138166943a0645f200844455858586f32127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f37607aeedf7e06c1f2a4042c3937cd0e8110b1a21bb3526f5363451afa4ae0ba67141c3448176c301d719882e67d8d04e9115f5b6ad5ab3ade0310445a5f314c20ab46a2ee4b03bf528f688e36689749dd550ed94c8cb4a9b12ff7e76eafef14cf846'
Test CA4	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d04030386610454d62af2abd57a84f24fd73dfbf1d42074b097549de2f50b349b321866734cad23c2b3705bdfc54b51e4e5d214aa4fc753fab4c7d443af476ff1c8efd4ce6aa9db8e0b65f0f27bd8df797f36dbb4cf6aade07ac21f5081fe22b497f1aefbce9f5f200844455858586f42127f4c1306082a8214004c0481185307bfffffffffffff5f25060103000902015f24060203000301085f376015369d871fb8a1fc6d1ff7c0f71289a07a04df5fe1df47da24e55cf475c47687239cf438668610a76c8536a0085dfb0e0d51cafe24138e4834e38c3fe4e05b387e28e0cd0f13db19ed01ac612946c31747d6a2616aa01ac8b69c7f3fff1f37dc5'

Test CA5	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d0403038661044844e9c051ddb7772cd5700eb9a192382f0b1fbf4a05ffca026510a8f3b70ef26437e8425643ba3b4995e6b1da512c444cc44c14533f253b2507adfb50544d5573823a4543156bbaf052e7dd54395284dded89b3829202212961c7b5e745ff95f200844455858586f52127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f376049b34f8f26d761ee83be427566481c5270e1a8a0bad98e3a1ef3492794031195e550f9358f6323feff87916ea731936b5cabb3c29e57f2b637b91e22ca5cefd6e92348319ab46ae6825acf4c6b332db6296f0d00cdce82b1eb65fc65b39f7aa'
Test CA6	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d0403038661045c9983c0ad853bc555a63f94b3245b688f2ac8de637f5426f9064d00b2f987776da4a4f7c445184dce6e76d2e5ef56a40f8d9e3c895790271923b561dfc675819457fcd9edc23093a16082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f37607c432ee2ec2640b13a6864ff201880c9392378983201c4abf8e1d55ed8a36c22ac7fd2b8bf8239f5f46ac1a17d1136af2dbd3035979890828d201872af34ce07a6b862f7d5c0ed4ba8dafa657443ecc04e1b31772911c5459cf0d4c4eb51582'
Test CA7	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d0403038661045949a568bc6e923f5120d246272947cc9ae33a3095b70a95e469c0ffble4763b121ece59e33f1275fb896f96e3e2c4752070bd62fae268693aa30bf995874302528074e32f3ba17f94433f0b71576f19f05de2589b395a8bd22409e1ddc93b3a5f200844455858586f72127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f376039cd3a5d6da66dfea52e514d06ef8186e4ddcdea98185764164a469f9d5f5a3315fb383b73c490d32d1b0424a9ca5c8923ac63aa349cad6620f7c8ea4611ee2724ccf5e68a90bfe7fa008931ae8f78401ee00fc37c57celf5a7c8a69792063a1'
Test CA8	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d04030386610479ebc9b3c7a373818959904d9154f63a4a08767955fdb23fa6225c39b70147ba5943facaa71e1852b13ba870beb27dbd4dffaeac0198f5bdc672b4da129778fb1eac47a75f79e4f4d55df706324d2b593ef00de4e2de19093f29f046906150a5f200844455858586f82127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f37608299ce56898da0b1b584130f11b03293f05278c9a3619b84f67fa12b9a972dd4001b013c78c2260a40987f813e76f6e73122767f4030c2832a7260b9d2e8a39de6164b35804735de29b65751006b084790ee68fdc08c792664ff4a89e3859cad'
Test CA9	'7f218201187f4e81b15f290170420844455858586002127f496d06082a8648ce3d0403038661041fcec13dbe6187eca90441f24f69059e97741ee761c934f0954fb16abf3070d80cf974910ce851d6fcbfa50a10b78de4c5586c538dccc5633152c32e158f22d6cf60b0562e509f1b65caf5fc43641cf952917c1d6eed4736d43029a5372bb85445f200844455858586f92127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f37602c787ce37d15b432af5cf70656900958c705d0ce025bd9a56075faed3ae7fc345f0c03b865787c487b1ff6730b50a7ee0b26245a89b31105641fe9c183297ac776147c5e572e271dbe3fd17d199a2bc1f94f7920e1c670daf2035108089e26e8'

Test:

(N211.210) K_externeWelt {K_Karte}

Dieser Prüfpunkt arbeitet mit folgenden Parametern:

- a. M_{CVC} = {TestCA0, TestCA1, ..., TestCA9}
- b. $PuK.RCA$ = / MF / DF.Auth / PuK.RCA_ELC384
- c. $PrK.Auth$ = / MF / DF.Auth / PrK.Auth_ELC384

(N211.220) K_externeWelt {K_Karte}

Die Performanzmessung wird wie folgt durchgeführt:

$(M_{Import}, M_{RoleCheck}, M_{SesKey}, M_{GetSecSta}) = PerformanceAuth(M_{CVC}, PuK.RCA, PrK.Auth)$

Testauswertung:

Es gilt:

$P_{Import,ELC384} = points(M_{Import}, T_{Import,ELC384})$

$P_{RoleCheck,ELC384} = points(M_{RoleCheck}, T_{RoleCheck,ELC384})$

$P_{\text{SesKey, ELC384}} = \text{points}(M_{\text{SesKey}}, T_{\text{SesKey, ELC384}})$

Hinweis (179): Das Ergebnis $M_{\text{GetSecSta}}$ wird hier nicht weiter verwendet, weil dieser Prüfpunkt bereits in B.8.11.1 betrachtet wird.

B.8.11.3 ELC 512

Es werden zehn Test CA benutzt, denen von der Root-CA (siehe PuK.RCA_ELC512) folgende CV-Zertifikate zugeordnet werden:

Test CA0	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d0403048681810495fc9b7543c2ff1ed4394027ad2acb94358fb6e82654e7157fd7c53c82bee4418c1f466c5c9ee3e7a10a5bb82947d97ae11690c784c216960a0ae68e300b449950ba3b9be3eb688e8c0b6d0b4fb03f5fbceacb21a00c31da9fb7f2c319e679d0d006739e18e80fca70c8702ae a4cd9d986fd84896e07ee851815170d7a5c95825f200844455858586f03127f4c1306082a8214004c0481185307bffffffffffff5f25060103000902015f24060203000301085f37818007b9a510c34a95f6fe3a5334db0145faa7678d524067a31b9817125724c5e278424dd6fa3a7b e20f4f762eb0938972b76195618445cb0c12a29b2f4693865b9d84d6d813641f4911f34b5d3 f7b9abccde60070a2fc85b2ac0c5d7ab44051b0f959399cf49a8beef2be13f934d3f976dca3 f77eb69e8499e4a28b5a3d0279a2da '
Test CA1	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d040304868181045f9a82a8f06b6ba10e32f9a21c002c4b96fde78bf47855d26aa4f8d52194756b7ec 537559d6a1c6e6a9d12c0634733279e862ce8c50cb68024898dc675d4b52359fdefffbf2d39 7d8f7e4f0365e0f13ddb633c2d6311205a268e502921249e429b73e1defdf5d834531d352b2 5a47162792d4c6096682ddbe36b11e54875689e5f200844455858586f13127f4c1306082a8214004c0481185307bffffffffffff5f25060103000902015f24060203000301085f3781804 e92e250b06ce31dlace367543fe3cba02b3d9b938397112a84f8c83a70f701acedefb94a047 892ce60d464e3cb22daf0calce2c964248a5b5619b7ea71f47f63e2343584b52290dbb46a3 c599ed7f5b2344acc3cf1bb9843999247061c0a76350585e0fb8ab14424b0fc5539c00d9999 1ce856a83bd1d57da069ea2b8c5a87 '
Test CA2	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d040304868181046038d73f8444a04b965a541af9247fef5d639be21793bb4bca616c773ac5dced51c ee5b36220fdbb2a4f320fbd5143fda7700cf17a2a0b49d03c9157ec9960309331754283ef76 f0db10cb599397cf8da23535857bb387ba097696f07020c75df500c6521b749fc53f38a57a5 62f18c2b5145ded45b6dd9ce7274a60bfe9d12e5f200844455858586f23127f4c1306082a8214004c0481185307bffffffffffff5f25060103000902015f24060203000301085f3781809 835527c0fa6f50056296af1dbe72c3f0836820f4be045d8f147048ba622ae785f58ead4118b 4d43f9c69899ca4c87d84d45c487ca309e0b50aa5d8513553d17941bd3cc90ee467347bdf0e 79e9db5dbb8ce16f5ad68dfb2227cdf42b0945dde03b87394b70c32c9c51763340e2f09fa8 5426552d3b0a0b8104c52cef8abf15 '
Test CA3	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d0403048681810499ca069ab8c0228c85b4c0d0bbb37d833d658b9ce81b0adc5b762224fd0011801f 321435df34677ccd21f6058832fc0b372df96b24ac4cb970681d381f5bc5c22d8a854adbe0b 9ddd15bb049589d1b1360816ced81c8cdf43106133b2612710a67024ae0e563f91e849aaf63 acc572dc48b9dcd6f87ce5cc203500dd0c638e55f200844455858586f33127f4c1306082a8214004c0481185307bffffffffffff5f25060103000902015f24060203000301085f3781805 f21aae86591351d5ddca7634eaa8a9f12e69f50a87d1d8a4835c24720098e66f1371262b4ca b80733a7e4ea28c9f180852eb19b4e2b0f3db0cb873ddfa92bf75acb2d072b9c810d415a7b6 73fd229ad16adeb23ecb2fb0fc7dca8a565fea391ff4d5b487fa3e5c37fb8d5775599586b04 4bae6589246b88c3215d05alc5b309 '
Test CA4	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d0403048681810431a80097158da05ecfea043eba55deead39c6e0a078c76e668a224ac4b7717213 b56521df9b4d77006aff6a5ffe1499070f33bb261a31f675631d6075df9be63d9083b253189 1a995accadbca7cf20dbc3db15d7b1bf545b5556d6fc7a889c0ff25c04bd2e186955d65625 7c936cc1e742b49de7eale8879b915e512cad545f200844455858586f43127f4c1306082a8214004c0481185307bffffffffffff5f25060103000902015f24060203000301085f3781809 99c42b069726e175b059480e2458ae101de5b3f842187ab8e11bc9b26651b729c8eba497fbc 56a6dd1259794ef530db11af9022aff968d237f050e4684935a89f49a69108492e71df26176 3d9ba1a46ae9e41bbe205906fb602f9f6b5d3407741f6f482daaa789c2e5131e95c09544664 7ea8dc75e40386248012c18a915e39 '

Test CA5	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d0403048681810451e931e921d6d7a771bc53e6c7770ed184817b5424a9918761046f34348d45a067713c04895e68c8cefc140e0a63b49703df483ba9a39f3fb06ee5e8eb4aba651cf33120c97df14dbfa34d13639ec70b11bbc6671e7e6f756705dd4365dc22e67bcad2e09b013ff9dacaadbecaef90069ca481cf9ab6e61f0f681e4ef91247845f200844455858586f53127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f37818091fea03f291142ff6b9942ef1b9c5fa191d53769ad1f156438815dd92438e6354b00ab9447f6ef79cedf8e51de1de8e4873e3982afb65261be556154059299a0058ebdb695d549213bb3d2a aaecc738f92cfd9c1e561e376b098ac4ec4dfc2d24075c62688681f366dc4e7ca5bd36c8cba61e8191aace6090ac83651b87bd309'
Test CA6	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d040304868181042edd3f3057e2cb122862bd3999e9368136d34330739a2a667720d1ea48b45fd7050b9645b099382c85759ae07074f76ea827c716d300700fd69b57dab282a4fa5bb4b1f62b19e99cccf5a3f0d24cd6937c31d28903dfcf5bc827bf8bd8c7502607f60b510d2c893663247b0b40d3f1d68bffc4e7f1aa27daeb745668ab3e0725f200844455858586f63127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f3781808c8203e3edca689def8a349e4cc0e3fad9aca951f63f4d0aa215a5333bd93b17795ccdbcf271f7fa243d60472f2cfe96be7b8ae7673a4b375436856d02d6df6c82afa534f533ba28316727478dabe5f4ecc215c442d4be6ac7a3e38890bbc8d758d8d75ada0d2c820c379b6b8744ecb9491d5010258a5cab4d929ba9e2f5cb9ee'
Test CA7	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d04030486818104140404028068847703b9ff2f2bd45927333e36da543533ff51d440cddb8d8dd526523524b89cd03320b8eb04593096c78cd0ab6ff4c3711fee04284d2e6bb003672d53f7111e05a14883e93482570667e62bd10c39c9999186d8486eba560abd2308275113861b6e74cf5968cc20bad34af21f74a4d54c985bcbcb3af9c88de05f200844455858586f73127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f3781809622b1a6b7be6087964877bf8d285e98a6d1f4defcb34eb48a34c7b0685f8c8da3a33d8bfff8b2b543c9567410da87ce8233f3d56d06b2d021beb30865b1d1193d87f835d12de3b5c89fefa45ca40378bc96d046f926c3d305ef888570f92d9c7ce53ffde6439a91b4e4be2a42208cfaadcb4661be54ab7b6300f2e1da47fde34a'
Test CA8	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d040304868181042416f09a534980a619c4a121687e82e5af1731f8de2c807c5eb00b4f458793b0ee2fa2b1e134d8aad1d1437a5fbee4c425e7b56d90d369f214de22ce41b1d5c8ec7d75a8b25464fe7fcc3ce4e713cb5d3addfc418ef2741a37a256e0b30f231688cc6c4705827e92f995e1cd9ade79129d32f570af100b7e95db030b88e7105f200844455858586f83127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f3781802717e550c4f907ad7a652904f960b3ea42eb03a689848556460e79fa60de3c8d1d8f2285b3a06a5cd17c12005ff4658194e819ac9cc715a0f56f4f7ac64e02f4073416c26a8a574a21c08262eda700036320d3492a80632c14a090a39533a8fcee7dd67c6f8a98adf04e5c1df03f14eb49ace6a6621aec24937fd56fad1be374'
Test CA9	'7f2182015b7f4e81d35f290170420844455858586003127f49818e06082a8648ce3d040304868181049bdab8d4c7e9fd163c9cfcfbf653909d2cef4032d1b0110990af18d14a538b30b3b7ff3abde1180ba8b07094eec56deda3e044611acab2ebc52f2da9bf8a08c894b1edfcf679b4c259b871df582588c81df4c9f0fe478bebf5d322f19ec843ae3aa704964775228ece5ad30927029f6f53cfc3c3663f48a4f4ddf7ad30fc65125f200844455858586f93127f4c1306082a8214004c0481185307bffffffffff5f25060103000902015f24060203000301085f378180a7cd37e35558fb279ef6acaf57aea6efc5d5d120e192dab917c7824d4fe4ea8b0a0b723add671c671704d667da2f9afc91522c06a175b3577370161e1b69bb399253b362661dab2cfde3c99091084a8b60b5c3132848e2c2962c1560ddc073143e2f8fec374d29f7f61c9cb621e15bfa00df40be5bb202f8f85554158ac7a965'

Test:

(N211.310) K_externerWelt {K_Karte}

Dieser Prüfpunkt arbeitet mit folgenden Parametern:

- a. M_{CVC} = {TestCA0, TestCA1, ..., TestCA9}
- b. $PuK.RCA$ = / MF / DF.Auth / PuK.RCA_ELC512
- c. $PrK.Auth$ = / MF / DF.Auth / PrK.Auth_ELC512

(N211.320) K_externerWelt {K_Karte}

Die Performanzmessung wird wie folgt durchgeführt:

$(M_{Import}, M_{RoleCheck}, M_{SesKey}, M_{GetSecSta}) = PerformanceAuth(M_{CVC}, PuK.RCA, PrK.Auth)$

Testauswertung:

Es gilt:

$$P_{Import,ELC512} = \text{points}(M_{Import}, T_{Import,ELC512})$$

$$P_{RoleCheck,ELC512} = \text{points}(M_{RoleCheck}, T_{RoleCheck,ELC512})$$

$$P_{SesKey,ELC512} = \text{points}(M_{SesKey}, T_{SesKey,ELC512})$$

Hinweis (180): Das Ergebnis $M_{GetSecSta}$ wird hier nicht weiter verwendet, weil dieser Prüfpunkt bereits in B.8.11.1 betrachtet wird.

B.8.11.4 RSA 2048

Dieses Kapitel ist absichtlich leer.

B.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 sowohl für RSA als auch für ELC durchlaufen, wobei ELC Schlüssel unterschiedlicher Länge geprüft werden.

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:		$(M_{Import}, M_{RoleCheck}, M_{SesKey}, M_{GetSecSta}) = \text{PerformanceAuth}(M_{CVC}, \text{PuK.RCA}, \text{PrK.Auth})$

Testvorbereitung:

Von jeder Test_CA aus M_{CVC} werden zehn CV-Zertifikate erzeugt. Insgesamt ergeben sich so einhundert CV-Zertifikate für Authentisierungsschlüssel.

(N211.510) K_externeWelt {K_eGK}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N211.520) K_externeWelt {K_eGK}

Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

(N211.600) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 0 bis 12 ausgeführt.

(N211.602) K_externeWelt {K_Karte}

Schritt 0: MSE Restore Kommando gemäß 14.9.9.1 mit *seNo* = 1. Dadurch werden alle Elemente aus der Liste *dfSpecificSecurityList* (siehe (N029.900)) entfernt. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N211.610) K_externeWelt {K_Karte}

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.

(N211.620) K_externeWelt {K_Karte}

Schritt 2: MSE Set Kommando gemäß 14.9.9.10, wobei als *keyRef* der Wert *keyIdentifier* aus *PuK.RCA* verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run2,i}$ bezeichnet.

(N211.630) K_externeWelt {K_Karte}

Schritt 3: Falls das im vorherigen Schritt selektierte Schlüsselobjekt ein

- ELC Schlüssel ist, wird ein PSO Verify Certificate Kommando gemäß 14.8.7.2 verwendet, wobei als Parameter *certificate* *CVC_Test_CAx* verwendet wird.
- Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run3,i}$ bezeichnet.

(N211.640) K_externeWelt {K_Karte}

Schritt 4: MSE Set Kommando gemäß 14.9.9.10, wobei als *keyRef* der Wert *CAR* aus *CVC_ICCy* verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run4,i}$ bezeichnet. Falls dieses Kommando nicht mit *NoError* beendet wird, fahre mit Schritt 2 fort, sonst mit Schritt 5.

(N211.650) K_externeWelt {K_Karte}

Schritt 5: PSO Verify Certificate Kommando gemäß 14.8.7.1, wobei als Parameter *certificate* *CVC_ICCy* verwendet wird. Die Laufzeit dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run5,i}$ bezeichnet.

(N211.660) K_externeWelt {K_Karte}

Schritt 6: MSE Set Kommando gemäß 14.9.9.5, 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 B.5.1 gemessen werden und wird mit $t_{Run6,i}$ bezeichnet.

(N211.670) K_externeWelt {K_Karte}

Schritt 7: Falls im vorherigen Schritt

- elcRoleCheck* verwendet wurde: GET CHALLENGE Kommando gemäß 14.9.4.2.

- b. Die Laufzeit dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run7,i}$ bezeichnet.

(N211.680) K_externeWelt {K_Karte}

Schritt 8: External Authenticate Kommando gemäß 14.7.1.1 und (N084.400){a, b}. Die Laufzeit dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run8,i}$ bezeichnet.

(N211.690) K_externeWelt {K_Karte}

Schritt 9: Abfrage eines Sicherheitsstatus:

- a. Falls M_{CVC} ELC Schlüssel enthält, dann werden folgende Schritte ausgeführt:

1. GET SECURITY STATUS KEY gemäß 14.7.3.3 mit $oid = oid_cvc_fl_ti$ und $cmdData = '0FFF...FF'$. Die Laufzeit dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run9.1,i}$ bezeichnet.
2. GET SECURITY STATUS KEY gemäß 14.7.3.3 mit $oid = oid_cvc_fl_ti$ und $cmdData = flagList$ aus CVC_ICCy. Die Laufzeit dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run9.2,i}$ bezeichnet.

- b. Es gilt: $t_{GetSecStat,i} = t_{Run9.1,i} + t_{Run9.2,i}$.

(N211.700) K_externeWelt {K_Karte}

Schritt 10: MSE Restore Kommando gemäß 14.9.9.1 mit $seNo = 1$. Dadurch werden alle Elemente aus der Liste $dfSpecificSecurityList$ entfernt. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N211.710) K_externeWelt {K_Karte}

Schritt 11: MSE Set Kommando gemäß 14.9.9.3, 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 B.5.1 gemessen werden und wird mit $t_{Run11,i}$ bezeichnet.

(N211.720) K_externeWelt {K_Karte}

Schritt 12: Aushandlung von Sessionkeys:

- a. Falls M_{CVC} ELC Schlüssel enthält, dann werden folgende Schritte ausgeführt:

1. GENERAL AUTHENTICATE gemäß 14.7.2.2.1, wobei als $keyRef$ der Wert CHR aus CVC_ICCy verwendet wird. Die Laufzeit dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run12.1,i}$ bezeichnet.
2. GENERAL AUTHENTICATE gemäß 14.7.2.2.2, wobei als $ephemeralPK_oponent$ die Antwort des vorherigen Kommandos verwendet wird. Die Laufzeit dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden und wird mit $t_{Run12.2,i}$ bezeichnet.

3. Es gilt: $t_{Run12,i} = t_{Run12.1,i} + t_{Run12.2,i}$.

Testauswertung:

Bei der Bearbeitung einer Schleifeniteration sind folgende Fälle denkbar:

- a. 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 4, 5, 6, 7, Die Laufzeiten der Schritte 4 und 5 werden summiert zu $t_{Import,i}$.

- b. 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 4, 2, 3, 4, 5, 6, 7, Die Laufzeiten der Schritte 4, 2, 3, 4 und 5 werden summiert zu $t_{Import,i}$.
- c. Die Laufzeiten der Schritte 6, 7 und 8 werden summiert zu $t_{RoleCheck,i}$.
- d. Die Laufzeiten der Schritte 11 und 12 werden aufsummiert zu $t_{SK,i}$.
- e. Die gemessenen Zeiten werden zu folgenden Tupeln zusammengefasst:

$$M_{Import} = (t_{Import,1}, t_{Import,2}, \dots, t_{Import,100})$$

$$M_{RoleCheck} = (t_{RoleCheck,1}, t_{RoleCheck,2}, \dots, t_{RoleCheck,100})$$

$$M_{SesKey} = (t_{SesKey,1}, t_{SesKey,2}, \dots, t_{SesKey,100})$$

$$M_{GetSecStat} = (t_{GetSecStat,1}, t_{GetSecStat,2}, \dots, t_{GetSecStat,100})$$

B.8.12 INTERNAL AUTHENTICATE zur Rollenauthentisierung

In diesem Abschnitt wird die Rollenauthentisierung mit privaten RSA und 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:

(N212.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N212.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

(N212.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS 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.

(N212.210) K_externeWelt {K_Karte}

Schritt 1: MSE Set Kommando gemäß 14.9.9.3, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = *elcRoleAuthentication* gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N212.220) K_externeWelt {K_Karte}

Schritt 2: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 4 ausgeführt.

(N212.230) K_externeWelt {K_Karte}

Schritt 3: Es wird eine Zufallszahl *token* = RAND(16) erzeugt.

(N212.240) K_externeWelt {K_Karte}

Schritt 4: INTERNAL AUTHENTICATE gemäß 14.7.4.1 und (N086.900){a, c}. Die Laufzeit $t_{Auth,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{RoleAuth,ELC256} = \text{points}(t_{Auth,1}, t_{Auth,2}, \dots, t_{Auth,100}, T_{roleAuth,ELC256})$$

$$P_{RoleAuth,ELC384} = \text{points}(t_{Auth,1}, t_{Auth,2}, \dots, t_{Auth,100}, T_{roleAuth,ELC384})$$

$$P_{RoleAuth,ELC512} = \text{points}(t_{Auth,1}, t_{Auth,2}, \dots, t_{Auth,100}, T_{roleAuth,ELC512})$$

Testnachbereitung:

Keine.

B.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:

(N213.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N213.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

(N213.130) K_externeWelt {K_Karte}

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.

(N213.140) K_externeWelt {K_Karte}

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:

(N213.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS 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.

(N213.220) K_externeWelt {K_Karte}

Schritt 1: MSE Set Kommando gemäß 14.9.9.9, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = signPSS gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N213.230) K_externeWelt {K_Karte}

Schritt 2: Die Testdurchführung MUSS eine innere Schleife 64-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 4 ausgeführt.

(N213.240) K_externeWelt {K_Karte}

Schritt 3: In der *i*-ten Schleifeniteration MUSS ein Oktettstring *dataToBeSigned* = RAND(*i*) erzeugt werden.

(N213.250) K_externeWelt {K_Karte}

Schritt 4: PSO Compute Digital Signature gemäß 14.8.2.1 und (N088.600)a. Die Laufzeit $t_{run,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{signPSS,2048} = \text{points}((t_{run,1}, t_{run,2}, \dots, t_{run,64}), T_{signPss,RSA2048})$$

$$P_{signPSS,3072} = \text{points}((t_{run,1}, t_{run,2}, \dots, t_{run,64}), T_{signPss,RSA3072})$$

Testnachbereitung:

Keine.

B.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:

(N214.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N214.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

Testdurchführung:

(N214.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS 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.

(N214.210) K_externeWelt {K_Karte}

Schritt 1: 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 dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N214.220) K_externeWelt {K_Karte}

Schritt 2: MSE Set Kommando gemäß 14.9.9.9, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = signECDSA gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N214.230) K_externeWelt {K_Karte}

Schritt 3: Die Testdurchführung MUSS eine innere Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 4 bis 6 ausgeführt.

(N214.240) K_externeWelt {K_Karte}

Schritt 4: In der i -ten Schleifeniteration MUSS ein Oktettstring *dataToBeSigned* = RAND(*domainParameter*. τ / 8) erzeugt werden.

(N214.250) K_externeWelt {K_Karte}

Schritt 5: PSO Compute Digital Signature gemäß 14.8.2.1 und (N088.600)c. Die

Laufzeit $t_{sign,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N214.260) K_externeWelt {K_Karte}

Schritt 6: PSO Verify Digital Signature gemäß 14.8.9.1 zur erfolgreichen Verifikation der im vorherigen Schritt erzeugten Signatur. Die Laufzeit $t_{verify,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{signECDSA,256} = \text{points}(t_{sign,1}, t_{sign,2}, \dots, t_{sign,100}, T_{signECDSA,ELC256})$$

$$P_{signECDSA,384} = \text{points}(t_{sign,1}, t_{sign,2}, \dots, t_{sign,100}, T_{signECDSA,ELC384})$$

$$P_{signECDSA,512} = \text{points}(t_{sign,1}, t_{sign,2}, \dots, t_{sign,100}, T_{signECDSA,ELC512})$$

$$P_{verifyECDSA,256} = \text{points}(t_{verify,1}, t_{verify,2}, \dots, t_{verify,100}, T_{verifyECDSA,ELC256})$$

$$P_{verifyECDSA,384} = \text{points}(t_{verify,1}, t_{verify,2}, \dots, t_{verify,100}, T_{verifyECDSA,ELC384})$$

$$P_{verifyECDSA,512} = \text{points}(t_{verify,1}, t_{verify,2}, \dots, t_{verify,100}, T_{verifyECDSA,ELC512})$$

Testnachbereitung:

Keine.

B.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:

(N215.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N215.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

Testdurchführung:

(N215.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS 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.

(N215.210) K_externeWelt {K_Karte}

Schritt 1: MSE Set Kommando gemäß 14.9.9.11, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = rsaDecipherOaep gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N215.220) K_externeWelt {K_Karte}

Schritt 2: Die Testdurchführung MUSS eine innere Schleife 190-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 5 ausgeführt.

(N215.230) K_externeWelt {K_Karte}

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 B.5.1 gemessen werden.

(N215.240) K_externeWelt {K_Karte}

Schritt 4: In der *i*-ten Schleifeniteration MUSS ein Oktettstring $M = \text{RAND}(i)$ erzeugt.

(N215.250) K_externeWelt {K_Karte}

Schritt 5: Falls *keyRef* gleich dem *keyIdentifier* von

- PrK.X509_RSA2048 ist, dann PSO Encipher gemäß 14.8.4.1 und (N091.700)c, wobei *PuK* der öffentlich Schlüssel ist, der in Schritt 3 erzeugt wurde und *algID* = *rsaEncipherOaep*. Die Laufzeit $t_{enc,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- PrK.X509_RSA3072 ist, dann wird *M* außerhalb des Prüflings äquivalent zu PSO Encipher gemäß 14.8.4.1 und (N091.700)c verschlüsselt, wobei *PuK* der öffentlich Schlüssel ist, der in Schritt 3 erzeugt wurde und *algID* = *rsaEncipherOaep*.

(N215.260) K_externeWelt {K_Karte}

Schritt 6: PSO Decipher gemäß 14.8.3.1 und (N090.300)b. Die Laufzeit $t_{dec,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{enc,2048} = \text{points}(t_{enc,1}, t_{enc,2}, \dots, t_{enc,190}, T_{enc,RSA2048})$$

$$P_{dec,2048} = \text{points}(t_{dec,1}, t_{dec,2}, \dots, t_{dec,190}, T_{dec,RSA2048})$$

$$P_{GAKP,2048} = \text{points}(t_{GAKP,1}, t_{GAKP,2}, \dots, t_{GAKP,190}, T_{GAKP,RSA2048})$$

$$P_{dec,3072} = \text{points}(t_{dec,1}, t_{dec,2}, \dots, t_{dec,190}, T_{dec,RSA3072})$$

$$P_{GAKP,3072} = \text{points}(t_{GAKP,1}, t_{GAKP,2}, \dots, t_{GAKP,190}, T_{GAKP,RSA3072})$$

Testnachbereitung:

Keine.

B.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:

(N216.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N216.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

Testdurchführung:

(N216.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine äußere Schleife durchlaufen, wobei *keyRef* jeden Wert *key/identifizier* 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.

(N216.210) K_externeWelt {K_Karte}

Schritt 1: MSE Set Kommando gemäß 14.9.9.11, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = *elcSharedSecretCalculation* gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N216.220) K_externeWelt {K_Karte}

Schritt 2: Die Testdurchführung MUSS eine innere Schleife 256-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 3 bis 5 ausgeführt.

(N216.230) K_externeWelt {K_Karte}

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 B.5.1 gemessen werden.

(N216.240) K_externeWelt {K_Karte}

Schritt 4: In der *i*-ten Schleifeniteration MUSS ein Oktettstring $M = \text{RAND}(i)$ erzeugt.

(N216.250) K_externeWelt {K_Karte}

Schritt 5: PSO Encipher gemäß 14.8.4.2, 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 B.5.1 gemessen werden.

(N216.260) K_externeWelt {K_Karte}

Schritt 6: PSO Decipher gemäß 14.8.3.2. Die Laufzeit $t_{dec,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{enc,256} = \text{points}((t_{enc,1}, t_{enc,2}, \dots, t_{enc,256}), T_{enc,ELC256})$$

$$P_{dec,256} = \text{points}((t_{dec,1}, t_{dec,2}, \dots, t_{dec,256}), T_{dec,ELC256})$$

$$P_{GAKP,256} = \text{points}((t_{GAKP,1}, t_{GAKP,2}, \dots, t_{GAKP,256}), T_{GAKP,ELC256})$$

$$P_{enc,384} = \text{points}((t_{enc,1}, t_{enc,2}, \dots, t_{enc,256}), T_{enc,ELC384})$$

$$P_{dec,384} = \text{points}((t_{dec,1}, t_{dec,2}, \dots, t_{dec,256}), T_{dec,ELC384})$$

$$P_{GAKP,384} = \text{points}((t_{GAKP,1}, t_{GAKP,2}, \dots, t_{GAKP,256}), T_{GAKP,ELC384})$$

$$P_{enc,512} = \text{points}((t_{enc,1}, t_{enc,2}, \dots, t_{enc,256}), T_{enc,ELC512})$$

$$P_{dec,512} = \text{points}((t_{dec,1}, t_{dec,2}, \dots, t_{dec,256}), T_{dec,ELC512})$$

$$P_{GAKP,512} = \text{points}((t_{GAKP,1}, t_{GAKP,2}, \dots, t_{GAKP,256}), T_{GAKP,ELC512})$$

Testnachbereitung:

Keine.

B.8.17 Selektieren von Ordnern und Logical Channel Reset

In diesem Abschnitt wird die Selektion eines Ordners und das Rücksetzen des Basiskanal betrachtet. Dieser Prüfpunkt arbeitet mit der Anwendung / MF / DF.Auth.

Testvorbereitung:

(N217.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

(N217.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 3 ausgeführt.

(N217.210) K_externeWelt {K_Karte}

Schritt 1: *currentFolder* MUSS gemäß 14.2.6.9 auf / MF / DF.Auth gesetzt werden. Die Laufzeit $t_{select,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N217.220) K_externeWelt {K_Karte}

Schritt 2: GET CHALLENGE wird gemäß 14.9.4.2 ausgeführt. Die Laufzeit $t_{rnd,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N217.230) K_externeWelt {K_Karte}

Schritt 3: Der Basiskanal MUSS gemäß Use Case aus 14.9.8.3 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 B.5.1 gemessen werden.

Testauswertung:

Es gilt:

$$P_{select_DF} = \text{points}((t_{select,1}, t_{select,2}, \dots, t_{select,100}), T_{select_DF})$$

$$P_{challenge} = \text{points}((t_{rnd,1}, t_{rnd,2}, \dots, t_{rnd,100}), T_{challenge})$$

$$P_{reset_Ch} = \text{points}((t_{reset,1}, t_{reset,2}, \dots, t_{reset,100}), T_{reset_Ch})$$

Testnachbereitung:

Keine.

B.8.18 MANAGE SECURITY ENVIRONMENT

In diesem Abschnitt wird die Selektion kryptographischer Objekte betrachtet. Dieser Prüfungspunkt arbeitet mit der Anwendung / MF / DF.IAS.

Testvorbereitung:

(N218.110) K_externeWelt {K_Karte}

Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N218.120) K_externeWelt {K_Karte}

Schritt 2: *currentFolder* MUSS auf / MF / DF.IAS gesetzt werden.

(N218.130) K_externeWelt {K_Karte}

Schritt 3: Es MUSS eine leere Menge $M_{Set} = \{\}$ erstellt werden.

Testdurchführung:

(N218.200) K_externeWelt {K_Karte}

Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.

(N218.210) K_externeWelt {K_Karte}

Schritt 1: MSE Set Kommando gemäß 14.9.9.3, wobei als *keyRef* PrK.X509_ELC256.*keyIdentifier* verwendet und *algId* = *elcRoleAuthentication* gesetzt wird. Die Laufzeit t_{Set} dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge M_{Set} hinzugefügt werden.

(N218.220) K_externeWelt {K_Karte}

Schritt 2: MSE Set Kommando gemäß 14.9.9.9, wobei als *keyRef* PrK.X509_ELC384.*keyIdentifier* verwendet und *algId* = *signECDSA* gesetzt wird. Die Laufzeit t_{Set} dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge M_{Set} hinzugefügt werden.

(N218.230) K_externeWelt {K_Karte}

Schritt 3: MSE Set Kommando gemäß 14.9.9.11, wobei als *keyRef* PrK.X509_ELC512.*keyIdentifier* verwendet und *algId* = *elcSharedSecretCalculation* gesetzt wird. Die Laufzeit t_{Set} dieses Kommandos MUSS gemäß B.5.1 gemessen und der Menge M_{Set} hinzugefügt werden.

(N218.240) K_externeWelt {K_Karte}

Schritt 4: MSE Restore Kommando gemäß 14.9.9.1, wobei *seNo* = 1 gesetzt wird. Die Laufzeit $t_{restore,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden

Testauswertung:

Es gilt:

$$P_{MSE_Set} = \text{points}(M_{Set}, T_{MSE_Set})$$

$$P_{MSE_Restore} = \text{points}(t_{restore,1}, t_{restore,2}, \dots, t_{restore,100}, T_{MSE_restore})$$

Testnachbereitung:

Keine.

B.8.19 GENERAL AUTHENTICATE, PACE

Dieser Prüfpunkt ist nur relevant, wenn Option_kontaktlose_Schnittstelle vorhanden ist.

In diesem Abschnitt wird die Etablierung mit einem symmetrischen Kartenverbindungsobjekt betrachtet. Dieser Prüfpunkt arbeitet mit dem Objekt / MF / DF.LCS / CAN_256.

Testvorbereitung:

(N219.110) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N219.120) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Schritt 2: *currentFolder* MUSS auf / MF / DF.LCS gesetzt werden.

Testdurchführung:

(N219.200) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 2 ausgeführt.

(N219.210) K_externeWelt {K_Karte}, Option_kontaktlose_Schnittstelle
Schritt 1: MSE Set Kommando gemäß 14.9.9.7 zur Auswahl von CAN_256. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N219.220) K_externeWelt {K_Karte}, 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 B.5.1 gemessen werden und zu $t_{PACE,i}$ addiert werden.

Testauswertung:

Es gilt:

$$P_{PACE} = \text{points}((t_{PACE,1}, t_{PACE,2}, \dots, t_{PACE,100}), T_{PACE})$$

Testnachbereitung:

Keine.

B.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:

(N220.110) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N220.120) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

(N220.200) K_externeWelt {K_Karte}, Option_Kryptobox
Die Testdurchführung MUSS 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.

(N220.210) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 1: MSE Set Kommandos gemäß 14.9.9.2 und 14.9.9.4, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4TC gesetzt wird. Die Laufzeiten dieser Kommandos sind für diesen Prüfpunkt irrelevant.

(N220.215) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 2: MSE Set Kommando gemäß 14.9.9.4, wobei als *keyRef* die Variable der äußeren Schleife verwendet wird und *algId* = aesSessionkey4TC gesetzt wird. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

(N220.220) K_externeWelt {K_Karte}, 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.

(N220.230) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 4: Es wird ein INTERNAL AUTHENTICATE Kommando gemäß 14.7.4.1 und (N086.902)a ausgeführt. Die Laufzeit $t_{IntAuth,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N220.240) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 5: Ein erfolgreiches EXTERNAL AUTHENTICATE Kommando 14.7.1.1 und (N084.402)a wird durchgeführt. Die Laufzeit $t_{ExtAuth,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N220.250) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 6: Die Ausführungszeiten werden wie folgt zusammengefasst:

$$t_{SK,i} = t_{IntAuth,i} + t_{ExtAuth,i}$$

(N220.260) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 7: MSE Restore Kommando gemäß 14.9.9.1 mit *seNo* = 1, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

Testauswertung:

Es gilt:

$$P_{SK4TC,AES128} = \text{points}((t_{SK128,1}, t_{SK128,2}, \dots, t_{SK128,100}), T_{SK4TC,AES128})$$

$$P_{SK4TC,AES192} = \text{points}((t_{SK192,1}, t_{SK192,2}, \dots, t_{SK192,100}), T_{SK4TC,AES192})$$

$$P_{SK4TC,AES256} = \text{points}((t_{SK256,1}, t_{SK256,2}, \dots, t_{SK256,100}), T_{SK4TC,AES256})$$

Testnachbereitung:

Keine.

B.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:

- (N221.110) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.
- (N221.120) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 2: *currentFolder* MUSS auf / MF / DF.Auth gesetzt werden.

Testdurchführung:

- (N221.200) K_externeWelt {K_Karte}, Option_Kryptobox
Die Testdurchführung MUSS 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.
- (N221.210) K_externeWelt {K_Karte}, 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.
- (N221.220) K_externeWelt {K_Karte}, 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.
- (N221.230) K_externeWelt {K_Karte}
Schritt 3: In der *i*-ten Schleifeniteration MUSS ein Oktettstring *cmdData* = RAND(*i*) erzeugt werden.
- (N221.240) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 4: Es wird ein PSO Encipher Kommando gemäß 14.8.4.5 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 B.5.1 gemessen werden.
- (N221.250) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 5: Es wird ein PSO Compute Cryptographic Checksum Kommando gemäß 14.8.1.1 und (N087.248)a durchgeführt, wobei als Kommandoparameter *data* das Chiffre *cipher* aus der Antwortnachricht des vorherigen Kommandos verwendet wird. Die Laufzeit $t_{computeCC,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.
- (N221.260) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 6: In der *i*-ten Schleifeniteration MUSS ein Oktettstring *rspData* = RAND(*i*) erzeugt werden. Diese Daten werden so verschlüsselt, dass das Chiffre *C* im folgenden PSO Decipher Kommando erfolgreich entschlüsselt wird. Zum Chiffre *C* wird ein MAC *mac* so berechnet, dass dieser im folgenden PSO Verify Cryptographic Checksum erfolgreich verifiziert wird.
- (N221.270) K_externeWelt {K_Karte}, Option_Kryptobox
Schritt 7: Es wird ein PSO Verify Cryptographic Checksum Kommando gemäß 14.8.8.1 durchgeführt, wobei im *inputTemplate* das Chiffre *C* in den MAC *mac* aus dem vorherigen Schritt enthält. Die Laufzeit $t_{verifyCC,i}$ dieses Kommandos in der *i*-ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N221.280) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 8: Es wird ein PSO Decipher Kommando gemäß 14.8.3.3 und (N090.302)a durchgeführt, wobei das Chiffre C aus Schritt 6 als Kommandoparameter verwendet wird. Die Laufzeit $t_{dec,i}$ dieses Kommandos in der i -ten Schleifeniteration MUSS gemäß B.5.1 gemessen werden.

(N221.290) K_externeWelt {K_Karte}, Option_Kryptobox

Schritt 9: MSE Restore Kommando gemäß 14.9.9.1 mit $seNo = 1$, wodurch Sicherheitszustände zurückgesetzt werden. Die Laufzeit dieses Kommandos ist für diesen Prüfpunkt irrelevant.

Testauswertung:

Durch die Messpunkt $(x, y) \rightarrow (i, t_{a,i})$ wird eine Ausgleichsgerade $y = m x + b$ gelegt.

Es gilt:

$$P_{compute128,b} = \text{points}((b, b), T_{compute128,b})$$

$$P_{compute128,m} = \text{points}((1000\ m, 1000\ m), T_{compute128,b})$$

$$P_{dec128,b} = \text{points}((b, b), T_{de128c,b})$$

$$P_{dec128,m} = \text{points}((1000\ m, 1000\ m), T_{dec128,b})$$

$$P_{enc128,b} = \text{points}((b, b), T_{enc128,b})$$

$$P_{enc128,m} = \text{points}((1000\ m, 1000\ m), T_{enc128,b})$$

$$P_{verify128,b} = \text{points}((b, b), T_{verify128,b})$$

$$P_{verify128,m} = \text{points}((1000\ m, 1000\ m), T_{verify128,b})$$

$$P_{compute192,b} = \text{points}((b, b), T_{compute192,b})$$

$$P_{compute192,m} = \text{points}((1000\ m, 1000\ m), T_{compute192,b})$$

$$P_{dec192,b} = \text{points}((b, b), T_{de192c,b})$$

$$P_{dec192,m} = \text{points}((1000\ m, 1000\ m), T_{dec192,b})$$

$$P_{enc192,b} = \text{points}((b, b), T_{enc192,b})$$

$$P_{enc192,m} = \text{points}((1000\ m, 1000\ m), T_{enc192,b})$$

$$P_{verify192,b} = \text{points}((b, b), T_{verify192,b})$$

$$P_{verify192,m} = \text{points}((1000\ m, 1000\ m), T_{verify192,b})$$

$$P_{compute256,b} = \text{points}((b, b), T_{compute256,b})$$

$$P_{compute256,m} = \text{points}((1000\ m, 1000\ m), T_{compute256,b})$$

$$P_{dec256,b} = \text{points}((b, b), T_{de256c,b})$$

$$P_{dec256,m} = \text{points}((1000\ m, 1000\ m) , T_{dec256,b})$$

$$P_{enc256,b} = \text{points}((b, b) , T_{enc256,b})$$

$$P_{enc256,m} = \text{points}((1000\ m, 1000\ m) , T_{enc256,b})$$

$$P_{verify256,b} = \text{points}((b, b) , T_{verify256,b})$$

$$P_{verify256,m} = \text{points}((1000\ m, 1000\ m) , T_{verify256,b})$$

Hinweis (181): 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.

B.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:

(N222.110) K_externeWelt {K_Karte}, Option_logische_Kanäle
Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

Testdurchführung:

(N222.200) K_externeWelt {K_Karte}, Option_logische_Kanäle
Die Testdurchführung MUSS eine Schleife 256-mal durchlaufen. In jedem Schleifendurchlauf wird Schritt 1 ausgeführt.

(N222.210) K_externeWelt {K_Karte}, Option_logische_Kanäle
Schritt 1: GET RANDOM wird gemäß 14.9.5.1 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 B.5.1 gemessen werden.

Testauswertung:

Durch die Messpunkt $(x, y) \rightarrow (i, t_{rnd,i})$ wird eine Ausgleichsgerade $y = m x + b$ gelegt.

Es gilt:

$$P_{Random,b} = \text{points}((b, b) , T_{Random,b})$$

$$P_{Random,m} = \text{points}((1000\ m, 1000\ m) , T_{Random,m})$$

Hinweis (182): 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.

B.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:

(N223.110) K_externeWelt {K_Karte}, Option_logische_Kanäle
Schritt 1: Der Prüfling MUSS gemäß B.5.2 aktiviert werden.

(N223.120) K_externeWelt {K_Karte}
Schritt 2: Es MÜSSEN folgende leere Mengen erstellt werden:

- a. $M_{Open} = \{\}$,
- b. $M_{Close} = \{\}$,
- c. $M_{RST} = \{\}$.

Testdurchführung:

(N223.200) K_externeWelt {K_Karte}, Option_logische_Kanäle
Die Testdurchführung MUSS eine Schleife 100-mal durchlaufen. In jedem Schleifendurchlauf werden die Schritte 1 bis 4 ausgeführt.

(N223.210) K_externeWelt {K_Karte}, Option_logische_Kanäle
Schritt 1: Zusätzlich zum Basiskanal MÜSSEN mittels MANAGE CHANNEL gemäß 14.9.8.1 drei weitere logische Kanäle geöffnet werden. Die Laufzeit dieser drei Kommandos MUSS gemäß B.5.1 gemessen und in die Menge M_{Open} eingestellt werden.

(N223.220) K_externeWelt {K_Karte}, Option_logische_Kanäle
Schritt 2: Die zusätzlich zum Basiskanal geöffneten logischen Kanäle MÜSSEN mittels MANAGE CHANNEL gemäß 14.9.8.2 geschlossen werden, wobei die Reihenfolge, in welcher die Kanäle geschlossen werden zufällig bestimmt wird.. Die Laufzeit dieser drei Kommandos MUSS gemäß B.5.1 gemessen und in die Menge M_{Close} eingestellt werden.

(N223.230) K_externeWelt {K_Karte}, Option_logische_Kanäle
Schritt 3: Zusätzlich zum Basiskanal MÜSSEN mittels MANAGE CHANNEL gemäß 14.9.8.1 drei weitere logische Kanäle geöffnet werden. Die Laufzeit dieser drei Kommandos MUSS gemäß B.5.1 gemessen und in die Menge M_{Open} eingestellt werden.

(N223.240) K_externeWelt {K_Karte}, Option_logische_Kanäle
Schritt 4: Die Applikationsebene MUSS mittels MANAGE CHANNEL gemäß 14.9.8.4 zurückgesetzt werden. Die Laufzeit dieses Kommandos MUSS gemäß B.5.1 gemessen und in die Menge M_{RST} eingestellt werden.

Testauswertung:

Es gilt:

$$P_{Open} = \text{points}(M_{Open}, T_{Open})$$

$$P_{Close} = \text{points}(M_{Close}, T_{Close})$$

$$P_{RST} = \text{points}(M_{RST}, T_{RST})$$

Testnachbereitung:

Keine.

B.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.

B.9.1 Attribute des Objektsystems

(N251.100) K_Personalisierung

Das Objektsystem gemäß (N019.900) MUSS folgende Attribute enthalten:

- a. Der Wert des Attributes *root* MUSS die Anwendung gemäß B.9.3 sein.
- b. Der Wert des Attributes *answerToReset* MUSS gemäß B.9.1.1 sein.
- c. Der Wert des Attributes *iccsn8* MUSS identisch zu den letzten acht Oktetten im *body* von / MF / EF.GDO sein.
- d. Das Attribut *applicationPublicKeyList* MUSS alle Schlüssel der folgenden Menge enthalten: {
 / MF / DF.Auth / PuK.RCA_ELC256,
 / MF / DF.Auth / PuK.RCA_ELC384,
 / MF / DF.Auth / PuK.RCA_ELC512}
- e. In *persistentCache* MUSS Platz für mindestens zehn CA-Schlüssel sein.
- f. Das Attribut *pointInTime* MUSS den Wert 2012.08.22 = '010200080202' besitzen.

B.9.1.1 Answer To Reset

(N251.200) K_Personalisierung

Für das Attribut *answerToReset* MUSS gelten:

- a. Der *answerToReset* MUSS den Vorgaben aus Tabelle 280 entsprechen.
- b. Der ATR SOLL ein TC1 Byte mit dem Wert 'FF' enthalten. In diesem Fall MUSS T0 auf den Wert 'Dx' gesetzt werden.
- c. Die Historical Bytes MÜSSEN gemäß [ISO/IEC 7816-4] codiert werden.

Tabelle 280: 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)

B.9.2 Allgemeine Festlegungen zu Attributstabellen

(N252.100) 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.
- Unterstützt ein COS weitere Kommandos oder Kommandovarianten, so KÖNNEN die Zugriffsbedingungen dafür herstellerspezifisch sein.

(N252.200) K_Personalisierung

SE abhängige Attribute:

- Alle Objekte MÜSSEN sich in SE#1 wie angegeben verwenden lassen.
- Objekte KÖNNEN in anderen SE verwendet werden
- Falls Objekte in anderen SE verwendbar sind, dann MÜSSEN sie dort dieselben Eigenschaften wie in SE#1 besitzen.

(N252.300) K_Personalisierung

Enthält eine Tabelle mit Ordnerattributen

- keinen *applicationIdentifier* (AID), so KANN diesem Ordner herstellerspezifisch ein beliebiger AID zugeordnet werden.
- einen oder mehrere AID, dann MUSS sich dieser Ordner mittels aller angegebenen AID selektieren lassen.
- keinen *fileIdentifier* (FID),
 - so DARF dieser Ordner NICHT mittels eines *fileIdentifier* aus dem Intervall gemäß 8.1.1 selektierbar sein, es sei denn, es handelt sich um den Ordner *root*, dessen optionaler *fileIdentifier* den Wert '3F00' besitzen MUSS.
 - so KANN diesem Ordner ein beliebiger *fileIdentifier* außerhalb des Intervalls gemäß 8.1.1 zugeordnet werden.

(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 8.1.2 selektieren lassen.

B.9.3 Root, die Wurzelapplikation

(N253.050) K_Personalisierung

Die Anwendung *root* MUSS die in Tabelle 281 dargestellten Attribute besitzen.

Tabelle 281: Attribute / MF

Attribute	Wert	Bemerkung
<i>objectType</i>	Ein Wert aus der Menge {Application, ADF}	
<i>applicationIdentifier</i>	'F000 0000 03'	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>fileIdentifier</i>	<i>objektType</i> = Application → kein <i>fileIdentifier</i> <i>objektType</i> = ADF → <i>fileIdentifier</i> = '3F 00'	
<i>children</i>	<ul style="list-style-type: none"> • / 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 	

B.9.3.1 / MF / EF.ATR

(N253.110) K_Personalisierung

Die Datei EF.ATR MUSS die in Tabelle 282 dargestellten Attribute besitzen.

Tabelle 282: Attribute / MF / EF.ATR

Attribute	Wert	Bemerkung
<i>objectType</i>	transparentes Elementary File	
<i>fileIdentifier</i>	'2F 01'	
<i>shortFileIdentifier</i>	'1D' = 29	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>flagTransactionMode</i>	False	
<i>flagChecksum</i>	True	
<i>numberOfOctet</i>	herstellerspezifisch	
<i>positionLogicalEndOfFile</i>	herstellerspezifisch	
<i>body</i>	'XX...YY'	siehe unten

Für das Attribut *body* gelten folgende Festlegungen:

(N253.120) K_Personalisierung

Der Oktettstring *body* MUSS DER-TLV codierte Datenobjekte (DO) enthalten, welche lückenlos hintereinander konkateniert werden MÜSSEN.

(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.

(N253.140) K_Personalisierung

In *body* KÖNNEN weitere DER-TLV codierte Datenobjekte enthalten sein.

B.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.

(N253.210) K_Personalisierung

Die Datei EF.DIR MUSS die in Tabelle 283 dargestellten Attribute besitzen.

(N253.220) K_Personalisierung

Für jede im Objektsystem vorhandene Anwendung MUSS ein eigener Rekord in EF.DIR enthalten sein, der diese Anwendung beschreibt.

Tabelle 283: Attribute / MF / EF.DIR

Attribute	Wert	Bemerkung
<i>objectType</i>	linear variables Elementary File	
<i>fileIdentifier</i>	'2F 00'	
<i>shortFileIdentifier</i>	'1E' = 30	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>flagTransactionMode</i>	True	
<i>flagChecksum</i>	True	
<i>maximumNumberOfRec.</i>	10 Rekord	
<i>maximumRecordLength</i>	36 Oktett	
<i>flagRecordLifeCycleStatus</i>	False	
<i>numberOfOctet</i>	'00BE' Oktett = 190 Oktett	
<i>recordList</i>		
Rekord 1	'61-L ₆₁ { 4F-05-F000000000 50-00 53-L ₅₃ -COS_Identifier }'	root, siehe B.9.3
Rekord 2	'61-L ₆₁ {4F-L _{4F} -AID}'	weitere Anwendungstemplates
...	...	

Hinweis (183): Der Oktettstring COS_Identifier wird von der gematik herstellerspezifisch festgelegt.

B.9.3.3 / MF / EF.GDO

(N253.310) K_Personalisierung

Die Datei EF.GDO MUSS die in Tabelle 284 dargestellten Attribute besitzen.

Tabelle 284: Attribute / MF / EF.GDO

Attribute	Wert	Bemerkung
<i>objectType</i>	transparentes Elementary File	
<i>fileIdentifier</i>	'2F 02'	
<i>shortFileIdentifier</i>	'02' = 2	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>flagTransactionMode</i>	False	
<i>flagChecksum</i>	True	
<i>numberOfOctet</i>	'000C' Oktett = 12 Oktett	
<i>positionLogicalEndOfFile</i>	12	
<i>body</i>	'5A-0A-80276...'	wird personalisiert

(N253.320) K_Personalisierung

In *body* MUSS genau ein DER-TLV codiertes Datenobjekt DO_ICCSN mit folgenden Eigenschaften enthalten sein:

- Tag = '5A' und Längensfeld = '0A'.
- Für das Wertfeld MUSS gelten:
 - Das erste Oktett MUSS den Major Industry Identifier (MII) mit dem Wert '80' enthalten.
 - Die nächsten drei Nibble MÜSSEN den Country Code Deutschlands mit dem Wert '276' enthalten.
 - Die nächsten fünf Nibble MÜSSEN den Issuer Identifier enthalten.
 - Die restlichen fünf Oktette MÜSSEN BCD codiert eine Seriennummer enthalten.

B.9.4 Anwendung für Authentisierungsprotokolle, DF.Auth

(N254.005) K_Personalisierung

Die Anwendung DF.Auth MUSS die in Tabelle 285 dargestellten Attribute besitzen.

Tabelle 285: Attribute / MF / DF.Auth

Attribute	Wert	Bemerkung
<i>objectType</i>	Ordner	
<i>applicationIdentifier</i>	'F000 0000 04'	
<i>fileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>children</i>	<ul style="list-style-type: none"> • / 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 	

Attribute	Wert	Bemerkung
	• / MF / DF.Auth / TC.AES256	

B.9.4.1 / MF/ DF.Auth / PrK.Auth_ELC256

(N254.010) K_Personalisierung

Der Schlüssel PrK.Auth_ELC256 MUSS die in Tabelle 286 dargestellten Attribute besitzen.

Tabelle 286: Attribute / MF / DF.Auth / PrK.Auth_ELC256

Attribute	Wert
<i>objectType</i>	privates Authentisierungsobjekt, ELC256
<i>keyIdentifier</i>	'11' = 17
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>privateKey</i>	<i>domainParameter</i> gemäß brainpoolP256r1, <i>privateElcKey.d</i> = '3b47afc75fafcf62ea3546efbe0b8d4a9295892e19acf562556441c34f374810'
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {elcRoleAuthentication, elcSessionkey4SM}

Hinweis (184): Der zugehörige öffentliche Punkt gemäß P2OS(...) lautet:

'040597cf45e706318271161744c8a2df37daf696c3feb0e3244de79d3a7472d8eb07d37f751e9494d4e9b12f9cea5a2105724f742d1e444a5553ec26da4e0556e9'.

B.9.4.2 / MF/ DF.Auth / PrK.Auth_ELC384

(N254.020) K_Personalisierung

Der Schlüssel PrK.Auth_ELC384 MUSS die in Tabelle 287 dargestellten Attribute besitzen.

Tabelle 287: Attribute / MF / DF.Auth / PrK.Auth_ELC384

Attribute	Wert
<i>objectType</i>	privates Authentisierungsobjekt, ELC384
<i>keyIdentifier</i>	'12' = 18
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>privateKey</i>	<i>domainParameter</i> gemäß brainpoolP384r1, <i>privateElcKey.d</i> = '5055f4c2260871efbf7eac69119597bb582be3210df5075629311a065d1328720f1982dc9d99ea1a4f3f4ad16c857ce8'
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {elcRoleAuthentication, elcSessionkey4SM}

Hinweis (185): Der zugehörige öffentliche Punkt gemäß P2OS(...) lautet:

'04808efcb3b017f52f46bafbc6ad5860d0f378ffe260edb0cbab2eff3f57c93c006958e0648e824c8fc211b9d33d564d70698419b30919ce79cdee85b06cd445146f254faa5df04debb56bf45355728b3e8a0d90560b002601d74c50f78d67aa6'.

B.9.4.3 / MF/ DF.Auth / PrK.Auth_ELC512

(N254.030) K_Personalisierung

Der Schlüssel PrK.Auth_ELC512 MUSS die in Tabelle 288 dargestellten Attribute besitzen.

Tabelle 288: Attribute / MF / DF.Auth / PrK.Auth_ELC512

Attribute	Wert
<i>objectType</i>	privates Authentisierungsobjekt, ELC512
<i>keyIdentifier</i>	'13' = 19
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>privateKey</i>	<i>domainParameter</i> gemäß brainpoolP512r1, <i>privateElcKey.d</i> = '4ef41d5494649f0a214dded44a77d617d41ca1f56795038c70e1b222ab6 d6d703c61dd6a6a2fa26e79db5848dedf1208ad2180afab576b23f23b310 84e03edc4'
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {elcRoleAuthentication, elcSessionkey4SM}

Hinweis (186): Der zugehörige öffentliche Punkt gemäß P2OS(...) lautet:

'0488ef0a9023aae89e687c9ab56b1e16d9dc5bdb7b4773dc1f883c4f257e917faba22c2f8f1c
20f9c283aa8197d70c592b2db4abcace1acf26a0f83f8ccd71665b88d5cb73df9b8ed006cd463
683e00a0bbc982c0f470543d4bf7190d13c8ecd21cc1b83db0421faa737d2fad35e0391d48ec
d11b25b3584fb691e66fd8786ab78'.

B.9.4.4 / MF/ DF.Auth / PrK.Auth_RSA2048

Dieses Kapitel ist absichtlich leer.

(N254.040) Dieser Punkt ist absichtlich leer.

B.9.4.5 / MF/ DF.Auth / PuK.RCA_ELC256

(N254.050) K_Personalisierung

Der Schlüssel PuK.RCA_ELC256 MUSS die in Tabelle 289 dargestellten Attribute besitzen.

Tabelle 289: Attribute / MF / DF.Auth / PuK.RCA_ELC256

Attribute	Wert
<i>objectType</i>	Öffentliches Signaturprüfobjekt, ELC256
<i>keyIdentifier</i>	'4445585858600112'
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>publicKey</i>	<i>domainParameter</i> gemäß brainpoolP256r1, P = '0434dc65068c3633671828f73adeb8fbc01952e3de3511823448f15afc28 e8ea3829be1ac9d4d10869ce5fa675275087603e4e9fc86ff3c127b9e94a 14c670164d'
<i>oid</i>	ecdsa-with-SHA256
<i>CHAT</i>	<ul style="list-style-type: none"> OID_{flags} = oid_cvc_fl_ti flagList = 'FF FFFF FFFF FFFF'
<i>expirationDate</i>	2023.04.15: YYMMDD = '0203 0004 0105'

Hinweis (187): Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d =

'74320bea049777bc663c27cfb2c488b7a49b05af348d9186f86a7196c2f6bb43'.

B.9.4.6 / MF/ DF.Auth / PuK.RCA_ELC384

(N254.060) K_Personalisierung

Der Schlüssel PuK.RCA_ELC384 MUSS die in Tabelle 290 dargestellten Attribute besitzen.

Tabelle 290: Attribute / MF / DF.Auth / PuK.RCA_ELC384

Attribute	Wert
<i>objectType</i>	Öffentliches Signaturprüfobjekt, ELC384
<i>keyIdentifier</i>	'4445585858600212'
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>publicKey</i>	<i>domainParameter</i> gemäß brainpoolP384r1, P = '043e84af62d1330a9a6f04a2791e0935e7cfe01eb8f7ae8d96eef77c036c 0569928e53700322a28454e647c85ea9c89c1d3051ce803c21e5c27a73b d8da121524d4679ded5144368523855e522cfe9a7d1ba31e691900cd027 40c399ffcc10f260'
<i>oid</i>	ecdsa-with-SHA384
<i>CHAT</i>	<ul style="list-style-type: none"> OIDflags = oid_cvc_fl_ti flagList = 'F F FFFF FFFF FFFF'
<i>expirationDate</i>	2023.04.15: YYMMDD = '0203 0004 0105'

*Hinweis (188): Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d =
'3dbf52550829cec527f91de05fc3d70b47c38d1c8504623174c912afc7941afbe104b9b3d34a
4eb0dd35452e2b67884a'.*

B.9.4.7 / MF/ DF.Auth / PuK.RCA_ELC512

(N254.070) K_Personalisierung

Der Schlüssel PuK.RCA_ELC512 MUSS die in Tabelle 291 dargestellten Attribute besitzen.

Tabelle 291: Attribute / MF / DF.Auth / PuK.RCA_ELC512

Attribute	Wert
<i>objectType</i>	Öffentliches Signaturprüfobjekt, ELC512
<i>keyIdentifier</i>	'4445585858600312'
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>publicKey</i>	<i>domainParameter</i> gemäß brainpoolP512r1, P = '043b5e694c154f4dd0812a91273869d382f5c81748ef5797416369b445f6 0dd2480d2400d1e7dd896b054ba615d8927fb10911e02753a22b1f6ec1b a18d0c1fdaa905712f33003bec6a51c179006e7b98f8d1d1219db49944f0 5602b46ac511570b136907081a5f93a0e000b7f335f76a7f8c18584fc7a7d 11fddc96844ff8cb5b'
<i>oid</i>	ecdsa-with-SHA512
<i>CHAT</i>	<ul style="list-style-type: none"> OIDflags = oid_cvc_fl_ti flagList = 'F F FFFF FFFF FFFF'
<i>expirationDate</i>	2023.04.15: YYMMDD = '0203 0004 0105'

*Hinweis (189): Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d =
'285927d1d7469dc69c83d25380dd51bbbd39cc69cf74219a53700eb44d1d0827130c6881ea
13836d454eae6972b5702b1da75e9ba39c6050539c5173129d2d0b'.*

B.9.4.8 / MF/ DF.Auth / PuK.RCA_RSA2048

Dieses Kapitel ist absichtlich leer.

(N254.080) Dieser Punkt ist absichtlich leer.

B.9.4.9 / MF / DF.Auth / SK.AES128

(N254.090) K_Personalisierung

Der Schlüssel SK.AES128 MUSS die in Tabelle 292 dargestellten Attribute besitzen.

Tabelle 292: Attribute / MF / DF.Auth / SK.AES128

Attribute	Wert	Bemerkung
<i>objectType</i>	symmetrisches Authentisierungsobjekt, AES128	
<i>keyIdentifier</i>	'02' = 2	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>encKey</i>	0102030405060708 090A0B0C0D0E0F10	
<i>macKey</i>	100F0E0D0C0B0A09 0807060504030201	
<i>numberScenario</i>	0	
<i>algorithmIdentifier</i>	aesSessionkey4SM	

B.9.4.10 / MF / DF.Auth / SK.AES192

(N254.100) K_Personalisierung

Der Schlüssel SK.AES192 MUSS die in Tabelle 293 dargestellten Attribute besitzen.

Tabelle 293: Attribute / MF / DF.Auth / SK.AES192

Attribute	Wert
<i>objectType</i>	Symmetrisches Authentisierungsobjekt, AES192
<i>keyIdentifier</i>	'03' = 3
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>encKey</i>	0102030405060708 090A0B0C0D0E0F10 1112131415161718
<i>macKey</i>	1817161514131211 100F0E0D0C0B0A09 0807060504030201
<i>numberScenario</i>	0
<i>algorithmIdentifier</i>	aesSessionkey4SM

B.9.4.11 / MF / DF.Auth / SK.AES256

(N254.110) K_Personalisierung

Der Schlüssel SK.AES256 MUSS die in Tabelle 294 dargestellten Attribute besitzen.

Tabelle 294: Attribute / MF / DF.Auth / SK.AES256

Attribute	Wert	Bemerkung
<i>objectType</i>	symmetrisches Authentisierungsobjekt, AES256	
<i>keyIdentifier</i>	'04' = 4	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>encKey</i>	0102030405060708 090A0B0C0D0E0F10 1112131415161718 191A1B1C1D1E1F20	
<i>macKey</i>	100F0E0D0C0B0A09 0807060504030201 201F1E1D1C1B1A19 1817161514131211	
<i>numberScenario</i>	0	
<i>algorithmIdentifier</i>	aesSessionkey4SM	

B.9.4.12 / MF / DF.Auth / TC.AES128

Dieses Objekt ist genau dann verfügbar, wenn Option_Kryptobox implementiert ist.

(N254.120) K_Personalisierung, Option_Kryptobox

Der Schlüssel TC.AES128 MUSS die in Tabelle 295 dargestellten Attribute besitzen.

Tabelle 295: Attribute / MF / DF.Auth / TC.AES128

Attribute	Wert	Bemerkung
<i>objectType</i>	symmetrisches Authentisierungsobjekt, AES128	
<i>keyIdentifier</i>	'05' = 5	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>encKey</i>	^0102030405060708 090A0B0C0D0E0F10^	
<i>macKey</i>	^100F0E0D0C0B0A09 0807060504030201^	
<i>algorithmIdentifier</i>	aesSessionkey4TC	

B.9.4.13 / MF / DF.Auth / TC.AES192

Dieses Objekt ist genau dann verfügbar, wenn Option_Kryptobox implementiert ist.

(N254.130) K_Personalisierung

Der Schlüssel TC.AES192 MUSS die in Tabelle 296 dargestellten Attribute besitzen.

Tabelle 296: Attribute / MF / DF.Auth / TC.AES192

Attribute	Wert
<i>objectType</i>	Symmetrisches Authentisierungsobjekt, AES192
<i>keyIdentifier</i>	'06' = 6
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>encKey</i>	^0102030405060708 090A0B0C0D0E0F10 1112131415161718^
<i>macKey</i>	^1817161514131211 100F0E0D0C0B0A09 0807060504030201^
<i>algorithmIdentifier</i>	aesSessionkey4TC

B.9.4.14 / MF / DF.Auth / TC.AES256

Dieses Objekt ist genau dann verfügbar, wenn Option_Kryptobox implementiert ist.

(N254.140) K_Personalisierung

Der Schlüssel TC.AES256 MUSS die in Tabelle 297 dargestellten Attribute besitzen.

Tabelle 297: Attribute / MF / DF.Auth / TC.AES256

Attribute	Wert	Bemerkung
<i>objectType</i>	symmetrisches Authentisierungsobjekt, AES256	
<i>keyIdentifier</i>	'07' = 7	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>encKey</i>	^0102030405060708 090A0B0C0D0E0F10 1112131415161718 191A1B1C1D1E1F20^	
<i>macKey</i>	^100F0E0D0C0B0A09 0807060504030201 201F1E1D1C1B1A19 1817161514131211^	
<i>algorithmIdentifier</i>	aesSessionkey4TC	

B.9.5 Anwendung für IAS Services, DF.IAS

(N255.050) K_Personalisierung

Die Anwendung DF.IAS MUSS die in Tabelle 298 dargestellten Attribute besitzen.

Tabelle 298: Attribute / MF / DF.IAS

Attribute	Wert	Bemerkung
<i>objectType</i>	Ordner	
<i>applicationIdentifier</i>	'F000 0000 05'	
<i>fileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>children</i>	<ul style="list-style-type: none"> • / 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 	

B.9.5.1 / MF/ DF.IAS / PrK.X509_ELC256

(N255.110) K_Personalisierung

Der Schlüssel PrK.X509_ELC256 MUSS die in Tabelle 299 dargestellten Attribute besitzen.

Tabelle 299: Attribute / MF / DF.IAS / PrK.X509_ELC256

Attribute	Wert
<i>objectType</i>	privates ELC Schlüsselobjekt, ELC256
<i>keyIdentifier</i>	'18' = 24
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>privateKey</i>	<i>domainParameter</i> gemäß brainpoolP256r1, <i>privateElcKey.d</i> = '295fcd16a620cb6fe34f7b0326e9b059c0b5229057c04ed9a7ed13a542967ee0'
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {elcRoleAuthentication, elcSharedSecretCalculation, signECDSA}

Hinweis (190): Der zugehörige öffentliche Punkt gemäß P2OS(...) lautet:

'04855978a5c0237c53404d6de6b9626145494feca14591dcfcbdb03850977598729ba33a5ae2dd510878c4edd669fbf6b55fddcbeffdbacfbec3b88e151fc78556'.

B.9.5.2 / MF/ DF.IAS / PrK.X509_ELC384

(N255.120) K_Personalisierung

Der Schlüssel PrK.X509_ELC384 MUSS die in Tabelle 300 dargestellten Attribute besitzen.

Tabelle 300: Attribute / MF / DF.IAS / PrK.X509_ELC384

Attribute	Wert
<i>objectType</i>	privates ELC Schlüsselobjekt, ELC384
<i>keyIdentifier</i>	'19' = 25
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>privateKey</i>	<i>domainParameter</i> gemäß brainpoolP384r1, <i>privateElcKey.d</i> = '6fbeeef714222b4a6ce80f035a00d80508cce63c73f4ad778db108f8e556f8a9f7625cce92fc6d24a2de0d8e998bee58'
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {elcSharedSecretCalculation, signECDSA}

Hinweis (191): Der zugehörige öffentliche Punkt gemäß P2OS(...) lautet:

'0435229000a37cc393cd04d8192ef25b910720a11453600a568b7f7adf3efebe984bc4893e3'

782aedfa9d4deeb602e757402e591fa8bc4cfe150e0b7a7504616f19dba2e7b55c6c11a89b60
61e0b2e831a7ccefd8b306400c2a0fe6e8bc5ebff84'.

B.9.5.3 / MF/ DF.IAS / PrK.X509_ELC512

(N255.130) K_Personalisierung

Der Schlüssel PrK.X509_ELC512 MUSS die in Tabelle 301 dargestellten Attribute besitzen.

Tabelle 301: Attribute / MF / DF.IAS / PrK.X509_ELC512

Attribute	Wert
<i>objectType</i>	privates ELC Schlüsselobjekt, ELC512
<i>keyIdentifier</i>	'1A' = 26
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>privateKey</i>	<i>domainParameter</i> gemäß brainpoolP512r1, <i>privateElcKey.d</i> = '2c24d847750f8321f875880deb583a7f4afdc8ebfd1fd6f587e6876d594bff baf284814019156c9efbafdac25ec426b6c842e82c5e4a657fee934c21b04 47810'
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {elcSharedSecretCalculation, signECDSA}

Hinweis (192): Der zugehörige öffentliche Punkt gemäß P2OS(...) lautet:

'043e4e30df5028cdc67a4aa3b95c4803a278e6ada8dd6d37ead2816fcf9507001d03929a692
137e376ee60c7318b3bb219d84f1bfb4429ed32b549c7544e006f226c787e46f2d0ed40e15e2
baec59b5f9bcd0dc8d189dc84c27c82aeb4de14183d95c5b2899f1530c8cdaddc75b262658
513d5716725745f101953d10f2e8a544'.

B.9.5.4 / MF/ DF.IAS / PrK.X509_RSA2048

(N255.140) K_Personalisierung

Der Schlüssel PrK.X509_RSA2048 MUSS die in Tabelle 302 dargestellten Attribute besitzen.

Tabelle 302: Attribute / MF / DF.IAS / PrK.X509_RSA2048

Attribute	Wert
<i>objectType</i>	privates RSA Schlüsselobjekt, RSA2048
<i>keyIdentifier</i>	'0B' = 11
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>modulusLength</i>	2048
<i>privateKey</i>	308204bd020100300d06092a864886f70d01010500048204a7308204a30201000282010100a20d2aa5d 0241c4cadd5db90509a3030a4f24f99c6ad2604ca96fd06986345a48eb74f1b9c7be26e18295fd58dd1a3 b6d715580ad27d2b5db93c168b8cdc261d92e12eee344624317a20d221cbbbcf1bd376d20126980f470f f2b48f65d686d9cc5e0f7383dcab0881a7de198f1a7b1a2717d1c03ad86973dcefd0425ed9bb98344549 eac891c9e25c3a02e873aafb5e2adc4afalc526081bd1f61e75c43fbcfbf135f52073d18d1a30c69d6aac 6956761c02a36d74de4fc6bb96c7529528a60628d99dea4abccb209da139398a68f563ab3595bd882b583 097a167b590932c9db5d47df32c10877bd5fe3cf46a31826259ef1329b5b99a2c87a40e78334590203010 0010282010034e0c8fecc394c46b51ea883ald97e4a1138c442b072c58a20b53e214dce6ee6306f9e4fab 333d82a13db6f8cf4b0df9d69b2f5c70acc95eced7cd9f81ba4071bc00e0b877b43f912981d62fad62915 7ec5f4eab7d7691cebl1a481f24ff9d0de9b9e723719520876ac22c0dac176713ae47fd5cc3363071d08dd 2728c1af83aa4f0209fe69356338dbff20a82cc55b8bd0096332538cc7f2357f78aff6ef73d78d618f46 773ccba18cc0b77f1258d395aecb3ba6af3e197111b08a2244cee1b5aa5d9cd76e50db7ebbabf61e36b92 51c77b3e7e91c12bc090c4c1bbd7f82952932f6f2a3c68f492de5d00af0be71214de359ae9a78740d21e8 6a4804ce5a50102818100c27b9a4daa01ef082498e84ced6d90bddd42c8682a7b359314cefedde7abb43a 098f93b75541d05b48d446c0fcd1c375fad17da4cf288dc6388c198739915edcbda3d6dc4c4df2ed74626 d006f73ab8acf1c8842cdd58a217a3972fab547fa377fcaab37dc0dc6799228e204705e8acd5f31419715 703c4dde4a87ae61311af902818100d54f682555c9966346e198495568238a74dc1dc0d821224311070a3 ffde4fc8b4d61c497e9fcdc40652b16defc1c6c595999cfd3a0588a5ab890789cf9d3cf92cfbd5186017 19d456173ef4701b62062572347fe2785d57652110c30c530d1af561d7c36104d0165047783bcdcd7db6ea 69a6ccdb05132a60c79fcd12b0adc6102818100ad83a7a4990486a5e6390230f83823c631ad4eae1910ba 38d27ce7c945d3dcc55718613b557695958c01d06a893c21ce960be8246dab09ea8b32f8b7b8c5e933da8 25dc548d2d6e5624c1a62240db843ed0ad0cb81677e88e5ca71b1a1a98f03629eecd119e25bebf33523029b e14188673901f23a00a795360818c8bb1cfdb9c90281800217c45ad16ba7e91371f52f8b01f98f4b3439a a81b45984b4ef0fbfaefb072b4ab811d8b7b7fe653758e3e18e31ad327739960f43f977ecc03118fcd627 a1c884137874c3c496414a12f2502da56721ce4e3f8b9daa36a83bdac63253b5a0e449d20aeed4cdca48d dc2c5c03875a4868265f379552886c9b047a61c97268210281806f99aa16030050301c99ea08e9d121ba 6a3f9e0a32cd265cfc00200855f640416e0980ea47a196ecb5c48f69e8c58aa4cd2289758f7fcd384ee74 4e6b87d727f8d4585c7f9924afd197469cf9910ba847841f9f67d024de97013e3c34c61b9f315d5f6b5cb 8bfe7316193c37b4d9a99a59b962fd3c75245227b42b325d33ccf3
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {rsaDecipherOaep, rsaDecipherPKCS1_V1_5, sign9796_2_DS2, signPSS}

B.9.5.5 / MF/ DF.IAS / PrK.X509_RSA3072

(N255.150) K_Personalisierung

Der Schlüssel PrK.X509_RSA3072 MUSS die in Tabelle 303 dargestellten Attribute besitzen.

Tabelle 303: Attribute / MF / DF.IAS / PrK.X509_RSA3072

Attribute	Wert
<i>objectType</i>	privates RSA Schlüsselobjekt, RSA3072
<i>keyIdentifier</i>	'0C' = 12
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>modulusLength</i>	3072
<i>privateKey</i>	308206fe020100300d06092a864886f70d01010500048206e8308206e40201000282018100c50ed1db3 704e8259e07d6d4433f4c2c45d0cf28a78dcb8512ebd4b09cd2a07f0ed22410b1188be8e5faf29c724266 24c4d73deee856179ee171199d5c09cebb23a42b2866477e26c8ff34c8fd5a53320fd82e81abfa649b9b 68bf876bd2b180d8886811bb6f48e64eded13286e39a74f61ff410cf1f6537ca2c73797a7a12507ad8b5a4 30ba1cd2c0c79cb550bf422ee4a75115050164c31a5784f1db433464dbfdd8fafca3fb3e12f519eb2f92a e1c67c88e5bd4fba8f10726317dd114ff64b644ba58b3a4cda9613219aa36d712300306b16f418c03f4ca 5f66ecf640808b98c547019d7b562a10b1f6d8ed9274a4e7159725ff5c20180434db6190235ae3ac905e4 a1981b3b488d11f78e1c586c85543be59bc2a313deb6914c14c8b4eb6838fd1a145805aaf026d22362fa 1c2b5f7e686bf1ec35c59c15201a93c1c549de51e79f32fe371c826fc99e2137268de8993bd0c27801237 02fe62db3bbac9b94d05b6764de631e8d1074df47a896a36fd46dc800966e3c70c09cb2de726f9d020301 000102820181008f37c11fd0b1f942f5c2fec8d0831d8a83c749f86326740728844bd57c74bc7efbb790c cadd98b4928a4a5553cd9b340f9eae8b14a1ce017fbee935041c742fdd1dc8cd1d55909a47f8073b646e49 39429ab48542c9c6064648fa3a6a7f69e6fb1942e376cc3cc12b0881bfa0f0d064888bf24e40e9f6a52f7 24320064f548f20fcd7a9d44631cda702c350ca9a8016cadcl61aa0eb3d30d4584dc33a74705ba5404ff 8112c2d6f3f3053e2c272c34eca2252a09e6166d6797e99348bcbf4c917a402f30a61b59e870773401d1 1e4345892ea98c64da681089a449dceaa2f4e4a806d08762c5084e3c040b688ddfad316889bfff178541eb 3a2781d37eed58fd24ac55928ae9db7f7419e6be825638df0b87f1a04912f1f0605e178d686e1864c7d5 4169d9df913cbf9db75a84a37640765037810f5aa74174c30ccc5aa2b9a895cbebf59fcd89a070a04 fddf83e0bd4f6e4dc540749eeb4291b553bd1703c8f585d439930b710069a2de833fdb42070d38e7c4be 6499d75699d52a7810281c100d570f346bc001fe691c060f81974d33d6c1a67b5c52b2891cb15a4faf22 df3342c3b417578cea4859a835b76f22c1ee83c50a6eca1f400b19586c82f84a2f8b1ab6489793cf24b4d 717dc65d73c7269577e298011c7cc3a21acd2c886d08cba8ecbd522f34552b9c4ff0c0ba0b495e2a0ae446 61f7106721c75b6abd33d3d228025da96a4ea8c8b77743330f4cefaac73d037b008afd2422b32be64c9b4 bf69a94e6a41fb057718a5200bbcf69819efb2b68669684ddee6c3ceb8f3afe76fde10281c100ce59964e 422745a2f7a850cfbcf666ec4948e5ce5b2cebb1fe927723b92235a94fbdac6bebe158b6efcb2cc3c5e25 8afca945658e82c966fc742c43c2d8b3f824fa395a01de92c6fd125b929fc2fc493daace672cb2f4ca940 5273a12ae749554440e4f34ea4b4bca973f98fe3116c765c4039391ca533273bd606a5611eebeb61a743 acc9525da1b12b4aa9db1bb74a6d690af77bf22105756883c35920ddd7651079c2b80272dc199dfee2bde 1bf3eb9406c8eaf32c19972f1c3064bc113d0281c0264622d9c44ca16bcbf27183c651435ed5bd966b59f 7fb686ae4c2b5174ac18b87a5605def2d2db5db9df643f681dd14d03c3d2ab0c3f9d003b37f81dcd430c2 58f6fe6f48764d5c837e9e773bbacf33740a136ffe83053c6f8d2ecd958937d28d702662de65b92820d794 ee6edaf325b87a87796899f3f5f27489553683fcbab7b104eaa44d86a6b77f83655c91074930811866ae2a 3e4a848e6c88d7a3e1734f34f46a07cd26670f5d625047331f371b09959b82793e184eb18bc1375730 10281c035619e931a5e85e8075219b69e0752a94fac3fcb19719bdcdf490b81ce5d345a704945f186ba 14c7602a42256594065048e49fbc00a611a4fdad78b208ab55714b4f3614c55f5ff9eaf4d14056419deb5 bf5c8a39f913f00eale779e4b12c311615ed49d449216675975e1c637ac1f691ab67c14079076eab1311f 934505994963fc307398942e08c356ceda431445aa90f7a8c9f0ff6e956b53e5d56ba17a0d890ba0f82a3 faf13f4ff20ba03b6688722adcd9f5340f89eae33d02e37f390281c1009f060963514741b1addc446bc2 b499010ad8a8893ae0c43b1acb0da212f80293a3a3c7b62f1d082ccf97ff0b8ccdc9caae64a293e5fd53 372cc7cc961158e81ba6ee86597c80d0ff890e7b7a3f5fce623548e1018b21c762a01d0aaa2aad8c2397 33fe7feb421e3caf25d347a9f45b78621ed808090e73e96acdb1da572d80f9e111125a86da90c7e7a81ec 9a0124d93d5943a8c77353dda990e85b5507336a73c1a4293a3698f0be81fc672d824510284108e860f c16feabce2b99666c1
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE: <i>setAlgorithmIdentifier</i> = {rsaDecipherOaep, rsaDecipherPKCS1_V1_5, sign9796_2_DS2, signPSS}

B.9.6 Anwendung für LCS Use Cases, DF.LCS

(N256.050) K_Personalisierung

Die Anwendung DF.LCS MUSS die in Tabelle 304 dargestellten Attribute besitzen.

Tabelle 304: Attribute / MF / DF.LCS

Attribute	Wert	Bemerkung
<i>objectType</i>	Ordner	
<i>applicationIdentifier</i>	'F000 0000 06'	
<i>fileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>children</i>	<ul style="list-style-type: none"> / 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 	

B.9.6.1 / MF / DF.LCS / CAN_256

Dieses Objekt ist genau dann verfügbar, wenn Option_kontaktlose_Schnittstelle implementiert ist.

(N256.100) K_Personalisierung, Option_kontaktlose_Schnittstelle

Der Schlüssel CAN_256 MUSS die in Tabelle 305 dargestellten Attribute besitzen.

Tabelle 305: Attribute / MF / DF.LCS / CAN_256

Attribute	Wert	Bemerkung
<i>objectType</i>	symmetrisches Kartenverbindungsobjekt	
<i>keyIdentifier</i>	'10' = 16	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>can</i>	123456	
<i>algorithmIdentifier</i>	für alle unterstützten SE: id-PACE-ECDH-GM-AES-CBC-CMAC-256	

B.9.6.2 / MF / DF.LCS / EF.LCS

(N256.200) K_Personalisierung

Die Datei EF.LCS MUSS die in Tabelle 306 dargestellten Attribute besitzen.

Tabelle 306: Attribute / MF / DF.LCS / EF.LCS

Attribute	Wert	Bemerkung
<i>objectType</i>	transparentes Elementary File	
<i>fileIdentifier</i>	'EF 01'	
<i>shortFileIdentifier</i>	'01' Oktett = 1 Oktett	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>flagTransactionMode</i>	False	
<i>flagChecksum</i>	False	
<i>numberOfOctet</i>	'0001' = 1	
<i>positionLogicalEndOfFile</i>	1	
<i>body</i>	'00'	

B.9.6.3 / MF / DF.LCS / PIN.LCS

(N256.300) K_Personalisierung

Das Passwortobjekt PIN.LCS MUSS die in Tabelle 307 dargestellten Attribute besitzen.

Tabelle 307: Attribute / MF / DF.LCS / PIN.LCS

Attribute	Wert	Bemerkung
<i>objectType</i>	Reguläres Passwortobjekt	
<i>pwdIdentifier</i>	'01' = 1	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>secret</i>	irrelevant	
<i>minimumLength</i>	4	
<i>maximumLength</i>	12	
<i>startRetryCounter</i>	3	
<i>retryCounter</i>	3	
<i>transportStatus</i>	Leer-PIN	
<i>flagEnabled</i>	True	
<i>startSsecList</i>	für alle unterstützten SE <i>startSsec</i> = unendlich	
<i>PUK</i>	12345678	
<i>pukUsage</i>	10	

B.9.6.4 / MF / DF.LCS / PrK.LCS

(N256.400) K_Personalisierung

Der Schlüssel PrK.LCS MUSS die in Tabelle 308 dargestellten Attribute besitzen.

Tabelle 308: Attribute / MF / DF.LCS / PrK.LCS

Attribute	Wert
<i>objectType</i>	privates ELC Schlüsselobjekt, ELC256
<i>keyIdentifier</i>	'11' = 17
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>privateKey</i>	<i>domainParameter</i> gemäß brainpoolP256r1, <i>privateElcKey.d</i> = '7625964fa8ca41673ac8c88a69fecfe18df90496c61f38d29310ccc4fc1484a1'
<i>keyAvailable</i>	True
<i>listAlgorithmIdentifier</i>	für alle unterstützten SE <i>setAlgorithmIdentifier</i> = {elcRoleAuthentication}

Hinweis (193): Der zugehörige öffentliche Punkt gemäß P2OS(...) lautet:

'044814679aa37c8c05f6c673a2b6cb621574dfb0b79835d3399e1bf1336eebdf7f67dc4670fae6fa60d456b40a0539472d5f5dcc2bf4da6872a9a3a13717dc2dfa'.

B.9.6.5 / MF / DF.LCS / PuK.LCS

(N256.500) K_Personalisierung

Der Schlüssel PuK.LCS MUSS die in Tabelle 309 dargestellten Attribute besitzen.

Tabelle 309: Attribute / MF / DF.LCS / PuK.LCS

Attribute	Wert
<i>objectType</i>	Öffentliches Signaturprüfobjekt, ELC256
<i>keyIdentifier</i>	'0102030405060708'
<i>lifeCycleStatus</i>	„Operational state (active)“
<i>publicKey</i>	<i>domainParameter</i> gemäß brainpoolP256r1, P = '049bbfb368605b45a6701d4e5fecbd52fbd43ee19340d802fa8792cac0a3 52f0e65fd6dd301fa7866a322d74c1cd0b634eac8a8e6fc600e1cd3361063 107d54b5b'
<i>oid</i>	ecdsa-with-SHA256
<i>CHAT</i>	<ul style="list-style-type: none"> OIDflags = oid_cvc_fl_ti flagList = 'BF FFFF FFFF FFFF'
<i>expirationDate</i>	2015.03.18: YYMMDD = '0105 0003 0108'

*Hinweis (194): Der zugehörige private Schlüssel besitzt das Attribut privateElcKey.d =
'715babdef8bc5828ce2ba351e4531a1d7ae8f64c1694ff9fb07e433231f678c6'.*

B.9.6.6 / MF / DF.LCS / SK.LCS

(N256.600) K_Personalisierung

Der Schlüssel SK.LCS MUSS die in Tabelle 310 dargestellten Attribute besitzen.

Tabelle 310: Attribute / MF / DF.LCS / SK.LCS

Attribute	Wert	Bemerkung
<i>objectType</i>	symmetrisches Authentisierungsobjekt, AES128	
<i>keyIdentifier</i>	'12' = 18	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>encKey</i>	'0102030405060708090A0B0C0D0E0F10'	
<i>macKey</i>	'100F0E0D0C0B0A090807060504030201'	
<i>numberScenario</i>	0	
<i>algorithmIdentifier</i>	aesSessionkey4SM	

B.9.7 Anwendung für Select EF Use Cases, DF.SelectEF

(N257.050) K_Personalisierung

Die Anwendung DF.SelectEF MUSS die in Tabelle 311 dargestellten Attribute besitzen.

Tabelle 311: Attribute / MF / DF.SelectEF

Attribute	Wert	Bemerkung
<i>objectType</i>	Ordner	
<i>applicationIdentifier</i>	'F000 0000 07'	
<i>fileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>children</i>	<ul style="list-style-type: none"> / MF / DF.SelectEF / EF.00 / MF / DF.SelectEF / EF.... / MF / DF.SelectEF / EF.99 	

B.9.7.1 / MF / DF.SelectEF / EF.xx

(N257.110) K_Personalisierung

Die Dateien EF.xx MÜSSEN die in Tabelle 312 dargestellten Attribute besitzen.

Tabelle 312: Attribute / MF / DF.SelectEF / EF.xx

Attribute	Wert	Bemerkung
<i>objectType</i>	transparentes Elementary File	
<i>fileIdentifier</i>	'EF 00' + <i>i</i> , mit <i>i</i> = 0, 1, 2, ..., 99	
<i>shortFileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>flagTransactionMode</i>	False	
<i>flagChecksum</i>	False	
<i>numberOfOctet</i>	'0001' = 1	
<i>positionLogicalEndOfFile</i>	1	
<i>body</i>	'00'	

Hinweis (195): Tabelle 312 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'}.

B.9.8 Anwendung für rekordorientierte Dateioperationen, DF.strukturiert

(N258.050) K_Personalisierung

Die Anwendung DF.strukturiert MUSS die in Tabelle 313 dargestellten Attribute besitzen.

Tabelle 313: Attribute / MF / DF.strukturiert

Attribute	Wert	Bemerkung
<i>objectType</i>	Ordner	
<i>applicationIdentifier</i>	'F000 0000 08'	
<i>fileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>children</i>	• / MF / DF.strukturiert / EF.strukturiert	

B.9.8.1 / MF / DF.strukturiert / EF.strukturiert

(N258.110) K_Personalisierung

Die Datei EF.strukturiert MUSS die in Tabelle 314 dargestellten Attribute besitzen.

Tabelle 314: Attribute / MF / DF.strukturiert / EF.strukturiert

Attribute	Wert	Bemerkung
<i>objectType</i>	linear variables Elementary File	
<i>fileIdentifier</i>	'EF 01'	
<i>shortFileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>flagTransactionMode</i>	True	
<i>flagChecksum</i>	True	
<i>maximumNumberOfRec.</i>	254	
<i>maximumRecordLength</i>	'FF' = 255	
<i>flagRecordLifeCycleStatus</i>	True	
<i>numberOfOctet</i>	'FD02' = 64.770	
<i>recordList</i>	leere Liste	

B.9.9 Anwendung für transparente Dateioperationen, DF.transparent

(N259.050) K_Personalisierung

Die Anwendung DF.transparent MUSS die in Tabelle 315 dargestellten Attribute besitzen.

Tabelle 315: Attribute / MF / DF.transparent

Attribute	Wert	Bemerkung
<i>objectType</i>	Ordner	
<i>applicationIdentifier</i>	'F000 0000 09'	
<i>fileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>children</i>	• / MF / DF.transparent / EF.transparent	

B.9.9.1 / MF / DF.transparent / EF.transparent

(N259.110) K_Personalisierung

Die Datei EF.transparent MUSS die in Tabelle 316 dargestellten Attribute besitzen.

Tabelle 316: Attribute / MF / DF.transparent / EF.transparent

Attribute	Wert	Bemerkung
<i>objectType</i>	transparentes Elementary File	
<i>fileIdentifier</i>	'EF 01'	
<i>shortFileIdentifier</i>	-	
<i>lifeCycleStatus</i>	„Operational state (active)“	
<i>flagTransactionMode</i>	True	
<i>flagChecksum</i>	True	
<i>numberOfOctet</i>	'8000' = 32.768	
<i>positionLogicalEndOfFile</i>	0	
<i>body</i>	irrelevant	

Anhang C – Domainparameter elliptischer Kurven (informativ)

Dieser Anhang enthält die Domainparameter der elliptischen Kurven, die gemäß dieser Dokumentenversion vom COS zu unterstützen sind. Die Object Identifier der hier dargestellten Kurven sind Tabelle 271 entnommen.

C.1 ansix9p256r1, OID = {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]:

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 (196): Gegenüber der Darstellung in [FIPS 186–4#D.1.2.3] ist zu beachten das gilt:
 $-3 \equiv a \bmod p$.

C.2 ansix9p384r1, OID = {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]:

p	'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000 FFFFFFFF'
a	'FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 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 FFFFFFFF FFFFFFFF C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973'
h	1

Hinweis (197): Gegenüber der Darstellung in [FIPS 186–4#D.1.2.3] ist zu beachten das gilt:
 $-3 \equiv a \bmod p$.

C.3 brainpoolP256r1, OID={1.3.36.3.3.2.8.1.1.7}='2B2403030208010107'

Die hier dargestellten Domainparameter wurden [RFC5639#3.4] entnommen:

<i>p</i>	'A9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377'
<i>a</i>	'7D5A0975FC2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9'
<i>b</i>	'26DC5C6CE94A4B44F330B5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B6'
<i>G</i>	in Koordinatenform: $x_g = '8BD2AEB9CB7E57CB2C4B482FFC81B7AFB9DE27E1E3BD23C23A4453BD9ACE3262'$ $y_g = '547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E545C1D54C72F046997'$
<i>n</i>	'A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A7'
<i>h</i>	1

C.4 brainpoolP384r1, OID={1.3.36.3.3.2.8.1.1.11}='2B240303020801010B'

Die hier dargestellten Domainparameter wurden [RFC5639#3.6] entnommen:

<i>p</i>	'8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B4 12B1DA197FB71123ACD3A729901D1A71874700133107EC53'
<i>a</i>	'7BC382C63D8C150C3C72080ACE05AFA0C2BEA28E4FB22787 139165EFBA91F90F8AA5814A503AD4EB04A8C7DD22CE2826'
<i>b</i>	'04A8C7DD22CE28268B39B55416F0447C2FB77DE107DCD2A6 2E880EA53EEB62D57CB4390295DBC9943AB78696FA504C11'
<i>G</i>	in Koordinatenform: $x_g = '1D1C64F068CF45FFA2A63A81B7C13F6B8847A3E77EF14FE3DB7FCAFE0CBD10E8E826E03436D646AAEF87B2E247D4AF1E'$ $y_g = '8ABE1D7520F9C2A45CB1EB8E95CFD55262B70B29FEEC5864E19C054FF99129280E4646217791811142820341263C5315'$
<i>n</i>	'8CB91E82A3386D280F5D6F7E50E641DF152F7109ED5456B3 1F166E6CAC0425A7CF3AB6AF6B7FC3103B883202E9046565'
<i>h</i>	1

C.5 brainpoolP512r1, OID={1.3.36.3.3.2.8.1.1.13}='2B240303020801010D'

Die hier dargestellten Domainparameter wurden [RFC5639#3.7] entnommen:

<i>p</i>	'AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA70330871 7D4D9B009BC66842AECDA12AE6A380E62881FF2F2D82C68528AA6056583A48F3'
<i>a</i>	'7830A3318B603B89E2327145AC234CC594CBDD8D3DF91610A83441CAEA9863BC 2DED5D5AA8253AA10A2EF1C98B9AC8B57F1117A72BF2C7B9E7C1AC4D77FC94CA'
<i>b</i>	'3DF91610A83441CAEA9863BC2DED5D5AA8253AA10A2EF1C98B9AC8B57F1117A7 2BF2C7B9E7C1AC4D77FC94CADC083E67984050B75EBAE5DD2809BD638016F723'
<i>G</i>	in Koordinatenform: $x_g = '81AEE4BDD82ED9645A21322E9C4C6A9385ED9F70B5D916C1B43B62EEF4D0098EFF3B1F78E2D0D48D50D1687B93B97D5F7C6D5047406A5E688B352209BCB9F822'$ $y_g = '7DDE385D566332ECC0EABFA9CF7822FDF209F70024A57B1AA000C55B881F8111B2DCDE49A45F485E5BCA4BD88A2763AED1CA2B2FA8F0540678CD1E0F3AD80892'$
<i>n</i>	'AADD9DB8DBE9C48B3FD4E6AE33C9FC07CB308DB3B3C9D20ED6639CCA70330870 553E5C414CA92619418661197FAC10471DB1D381085DDADB58796829CA90069'
<i>h</i>	1

Anhang D – Erläuterungen zu Wertebereichen (informativ)

Hinweis (198): 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. Falls 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.

Anhang E – Verzeichnisse (informativ)

E.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

E.2 Glossar

Begriff	Erläuterung
ephemer	Nur für kurze Zeit bestehend, flüchtig, ohne bleibende Bedeutung (siehe http://de.wiktionary.org/wiki/ephemer)
<i>flagList</i>	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: <ul style="list-style-type: none"> 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.

E.3 Abbildungsverzeichnis

Abbildung 1: Message Sequence Chart für die Kommandobearbeitung	131
Abbildung 2: Zeitlicher Ablauf eines Roll-Back-Kommandos	165
Abbildung 3: Kommunikationsmodell Gegenstelle, Steuersoftware und COS	389
Abbildung 4: Sequenzdiagramm zur externen Authentisierung	390
Abbildung 5: Sequenzdiagramm zur internen Authentisierung	392
Abbildung 6: Sequenzdiagramm symmetrische Sessionkey-Aushandlung.....	393
Abbildung 7: Sequenzdiagramm PACE Authentisierung	395
Abbildung 8: Sequenzdiagramm RSA-Sessionkey-Aushandlung.....	396
Abbildung 9: Sequenzdiagramm ELC-Sessionkey-Aushandlung	399
Abbildung 10: Sequenzdiagramm zur Übertragung von Sessionkeys	400
Abbildung 11: Komponentendiagramm Performanzmessplatz	407
Abbildung 12: Graphische Darstellung von $P_{gesamt_einfach}$	416
Abbildung 13 - Synchrone Kommunikation zw. CMS und Smartcard	492
Abbildung 14 - Asynchrone Kommunikation zw. CMS und Smartcard	492
Abbildung 15: Zusammenhang Schlüsselspeicher	502

E.4 Tabellenverzeichnis

Tabelle 1: Präfixe, die auf Vielfachen von Zehnerpotenzen beruhen:.....	23
---	----

Tabelle 2: Präfixe, die auf Vielfachen von Zweierpotenzen beruhen:	24
Tabelle 3: Liste der Komponenten, an welche dieses Dokument Anforderungen stellt.....	25
Tabelle 4: Liste der Schlüsselparameter eines RSA-Schlüssels	46
Tabelle 5: CV-Zertifikat einer CA mit CPI = '21', SHA-256	68
Tabelle 6: CV-Zertifikat zur Authentisierung mit CPI = '22', SHA-256.....	68
Tabelle 7: Liste der Domainparameter einer elliptischen Kurve.....	74
Tabelle 8: Aufheben des Transportschutzes	77
Tabelle 9: Codierung des CLA-Bytes für Kanalnummern kleiner vier	132
Tabelle 10: Codierung des CLA-Bytes für Kanalnummern größer gleich vier	133
Tabelle 11: Case 1 Kommando-APDU.....	136
Tabelle 12: Case 1 Antwort-APDU.....	136
Tabelle 13: Case 2 Short Kommando-APDU	137
Tabelle 14: Case 2 Extended Kommando-APDU	137
Tabelle 15: Case 2 Antwort-APDU.....	138
Tabelle 16: Case 3 Short Kommando-APDU	138
Tabelle 17: Case 3 Extended Kommando-APDU	139
Tabelle 18: Case 3 Antwort-APDU.....	140
Tabelle 19: Case 4 Short Kommando-APDU	140
Tabelle 20: Case 4 Extended Kommando-APDU	141
Tabelle 21: Case 4 Antwort-APDU.....	142
Tabelle 22: Kommandos, alphabetisch	164
Tabelle 23: Kommandos, numerisch.....	164
Tabelle 24: ACTIVATE aktuelles File.....	167
Tabelle 25: ACTIVATE privates oder symmetrisches Schlüsselobjekt.....	167
Tabelle 26: ACTIVATE öffentliches Schlüsselobjekt.....	168
Tabelle 27: ACTIVATE Passwortobjekt	168
Tabelle 28: ACTIVATE Antwort-APDU im Erfolgsfall	169
Tabelle 29: ACTIVATE Antwort-APDU im Fehlerfall	169
Tabelle 30: DEACTIVATE aktuelles File.....	171
Tabelle 31: DEACTIVATE privates oder symmetrisches Schlüsselobjekt.....	171
Tabelle 32: DEACTIVATE öffentliches Schlüsselobjekt.....	172
Tabelle 33: DEACTIVATE Passwortobjekt	173
Tabelle 34: DEACTIVATE Antwort-APDU im Erfolgsfall	173
Tabelle 35: Deactivate Antwort-APDU im Fehlerfall	173
Tabelle 36: DELETE aktuelles File.....	175
Tabelle 37: DELETE privates oder symmetrisches Schlüsselobjekt.....	176

Tabelle 38: DELETE öffentliches Schlüsselobjekt	176
Tabelle 39: DELETE Passwortobjekt	177
Tabelle 40: DELETE Antwort-APDU im Erfolgsfall	177
Tabelle 41: Delete Antwort-APDU im Fehlerfall.....	177
Tabelle 42: LOAD APPLICATION mit Command Chaining	180
Tabelle 43: LOAD APPLICATION ohne Command Chaining	181
Tabelle 44: LOAD APPLICATION Antwort-APDU im Erfolgsfall	181
Tabelle 45: LOAD APPLICATION Antwort-APDU im Fehlerfall	181
Tabelle 46: SELECT, kein AID, first occurrence, keine Antwortdaten.....	185
Tabelle 47: SELECT, kein AID, first occurrence, Antwortdaten mit FCP	185
Tabelle 48: SELECT, kein AID, next occurrence, keine Antwortdaten.....	186
Tabelle 49: SELECT, kein AID, next occurrence, Antwortdaten mit FCP	187
Tabelle 50: SELECT, AID, first occurrence, keine Antwortdaten	187
Tabelle 51: SELECT, AID, first occurrence, Antwortdaten mit FCP.....	188
Tabelle 52: SELECT, AID, next occurrence, keine Antwortdaten	189
Tabelle 53: SELECT, AID, next occurrence, Antwortdaten mit FCP	190
Tabelle 54: SELECT, DF oder ADF mit FID, keine Antwortdaten	190
Tabelle 55: SELECT, DF oder ADF mit FID, Antwortdaten mit FCP.....	191
Tabelle 56: SELECT, höhere Ebene keine Antwortdaten	192
Tabelle 57: SELECT, höhere Ebene, Antwortdaten mit FCP.....	192
Tabelle 58: SELECT, EF mit FID, keine Antwortdaten	193
Tabelle 59: SELECT, EF mit FID, Antwortdaten mit FCP	194
Tabelle 60: SELECT, Kommandoparameter im Überblick.....	194
Tabelle 61: SELECT Antwort-APDU im Erfolgsfall	195
Tabelle 62: SELECT Antwort-APDU im Fehlerfall	195
Tabelle 63: TERMINATE CARD USAGE	198
Tabelle 64: TERMINATE CARD USAGE Antwort-APDU im Erfolgsfall.....	198
Tabelle 65: Terminate Card Usage Antwort-APDU im Fehlerfall	199
Tabelle 66: TERMINATE DF aktueller Ordner	200
Tabelle 67: TERMINATE DF Antwort-APDU im Erfolgsfall	200
Tabelle 68: TERMINATE DF Antwort-APDU im Fehlerfall	201
Tabelle 69: TERMINATE aktuelle Datei	202
Tabelle 70: TERMINATE privates oder symmetrisches Schlüsselobjekt	203
Tabelle 71: TERMINATE öffentliches Schlüsselobjekt	203
Tabelle 72: TERMINATE Passwortobjekt.....	204
Tabelle 73: TERMINATE Antwort-APDU im Erfolgsfall.....	204

Tabelle 74: TERMINATE Antwort-APDU im Fehlerfall.....	204
Tabelle 75: ERASE BINARY, logical EOF unverändert, ohne <i>shortFileIdentifier</i>	206
Tabelle 76: ERASE BINARY, logical EOF unverändert, mit <i>shortFileIdentifier</i>	207
Tabelle 77: ERASE BINARY Antwort-APDU im Erfolgsfall.....	207
Tabelle 78: ERASE BINARY Antwort-APDU im Fehlerfall.....	207
Tabelle 79: READ BINARY ohne <i>shortFileIdentifier</i>	210
Tabelle 80: READ BINARY mit <i>shortFileIdentifier</i>	211
Tabelle 81: READ BINARY Antwort-APDU im Erfolgsfall	211
Tabelle 82: READ BINARY Antwort-APDU im Fehlerfall	211
Tabelle 83: SET LOGICAL EOF ohne <i>shortFileIdentifier</i>	214
Tabelle 84: SET LOGICAL EOF mit <i>shortFileIdentifier</i>	214
Tabelle 85: SET LOGICAL EOF Antwort-APDU im Erfolgsfall.....	214
Tabelle 86: SET LOGICAL EOF Antwort-APDU im Fehlerfall	214
Tabelle 87: UPDATE BINARY ohne <i>shortFileIdentifier</i>	217
Tabelle 88: UPDATE BINARY mit <i>shortFileIdentifier</i>	218
Tabelle 89: UPDATE BINARY Antwort-APDU im Erfolgsfall.....	218
Tabelle 90: UPDATE BINARY Antwort-APDU im Fehlerfall.....	218
Tabelle 91: WRITE BINARY ohne <i>shortFileIdentifier</i>	221
Tabelle 92: WRITE BINARY mit <i>shortFileIdentifier</i>	222
Tabelle 93: WRITE BINARY Antwort-APDU im Erfolgsfall.....	222
Tabelle 94: WRITE BINARY Antwort-APDU im Fehlerfall.....	222
Tabelle 95: ACTIVATE RECORD, ein Rekord, ohne <i>shortFileIdentifier</i>	225
Tabelle 96: ACTIVATE RECORD, ein Rekord, mit <i>shortFileIdentifier</i>	226
Tabelle 97: ACTIVATE RECORD, alle Rekords ab P1, ohne <i>shortFileIdentifier</i>	226
Tabelle 98: ACTIVATE RECORD, alle Rekords ab P1, mit <i>shortFileIdentifier</i>	227
Tabelle 99: ACTIVATE RECORD Antwort-APDU im Erfolgsfall	227
Tabelle 100: ACTIVATE RECORD Antwort-APDU im Fehlerfall	227
Tabelle 101: APPEND RECORD, ohne <i>shortFileIdentifier</i>	230
Tabelle 102: APPEND RECORD, mit <i>shortFileIdentifier</i>	230
Tabelle 103: APPEND RECORD Antwort-APDU im Erfolgsfall.....	231
Tabelle 104: APPEND RECORD Antwort-APDU im Fehlerfall.....	231
Tabelle 105: DEACTIVATE RECORD, ein Rekord, ohne <i>shortFileIdentifier</i>	234
Tabelle 106: DEACTIVATE RECORD, ein Rekord, mit <i>shortFileIdentifier</i>	235
Tabelle 107: DEACTIVATE RECORD, alle Rekords ab P1, ohne <i>shortFileIdentifier</i>	235
Tabelle 108: DEACTIVATE RECORD, alle Rekords ab P1, mit <i>shortFileIdentifier</i>	236
Tabelle 109: DEACTIVATE RECORD Antwort-APDU im Erfolgsfall	236

Tabelle 110: DEACTIVATE RECORD Antwort-APDU im Fehlerfall	236
Tabelle 111: DELETE RECORD, ohne <i>shortFileIdentifier</i>	239
Tabelle 112: DELETE RECORD, mit <i>shortFileIdentifier</i>	240
Tabelle 113: DELETE RECORD Antwort-APDU im Erfolgsfall	240
Tabelle 114: DELETE RECORD Antwort-APDU im Fehlerfall	240
Tabelle 115: ERASE RECORD, ohne <i>shortFileIdentifier</i>	242
Tabelle 116: ERASE RECORD, mit <i>shortFileIdentifier</i>	243
Tabelle 117: ERASE RECORD Antwort-APDU im Erfolgsfall.....	243
Tabelle 118:ERASE RECORD Antwort-APDU im Fehlerfall.....	243
Tabelle 119: READ RECORD ohne <i>shortFileIdentifier</i>	246
Tabelle 120: READ RECORD mit <i>shortFileIdentifier</i>	247
Tabelle 121: READ RECORD Antwort-APDU im Erfolgsfall.....	247
Tabelle 122: READ RECORD Antwort-APDU im Fehlerfall.....	247
Tabelle 123: SEARCH RECORD ohne <i>shortFileIdentifier</i>	250
Tabelle 124: SEARCH RECORD mit <i>shortFileIdentifier</i>	251
Tabelle 125: SEARCH RECORD Antwort-APDU im Erfolgsfall.....	251
Tabelle 126: SEARCH RECORD Antwort-APDU im Fehlerfall.....	251
Tabelle 127: UPDATE RECORD, ohne <i>shortFileIdentifier</i>	254
Tabelle 128: UPDATE RECORD mit <i>shortFileIdentifier</i>	255
Tabelle 129: UPDATE RECORD Antwort-APDU im Erfolgsfall.....	255
Tabelle 130: UPDATE RECORD Antwort-APDU im Fehlerfall.....	255
Tabelle 131: CHANGE REFERENCE DATA mit altem und neuem Benutzergeheimnis	260
Tabelle 132: CHANGE REFERENCE DATA, nur neues Benutzergeheimnis	260
Tabelle 133: CHANGE REFERENCE DATA Antwort-APDU im Erfolgsfall	261
Tabelle 134: CHANGE REFERENCE DATA Antwort-APDU im Fehlerfall.....	261
Tabelle 135: DISABLE VERIFICATION REQUIREMENT mit Verifikationsdaten.....	264
Tabelle 136: DISABLE VERIFICATION REQUIREMENT ohne Verifikationsdaten	264
Tabelle 137: DISABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall	264
Tabelle 138: DISABLE VERIFICATION REQUIREMENT Antwort-APDU im Fehlerfall	264
Tabelle 139: ENABLE VERIFICATION REQUIREMENT mit Verifikationsdaten	267
Tabelle 140: ENABLE VERIFICATION REQUIREMENT ohne Verifikationsdaten	267
Tabelle 141: ENABLE VERIFICATION REQUIREMENT Antwort-APDU im Erfolgsfall	267
Tabelle 142: ENABLE VERIFICATION REQUIREMENT Antwort-APDU im Fehlerfall	268
Tabelle 143: GET PIN STATUS.....	270
Tabelle 144: GET PIN STATUS Antwort-APDU im Erfolgsfall.....	270
Tabelle 145: GET PIN STATUS Antwort-APDU im Fehlerfall.....	270

Tabelle 146: RESET RETRY COUNTER, mit PUK, mit <i>newSecret</i>	273
Tabelle 147: RESET RETRY COUNTER, mit PUK, ohne <i>newSecret</i>	273
Tabelle 148: RESET RETRY COUNTER, ohne PUK, mit <i>newSecret</i>	274
Tabelle 149: RESET RETRY COUNTER, ohne PUK, ohne <i>newSecret</i>	274
Tabelle 150: RESET RETRY COUNTER Antwort-APDU im Erfolgsfall	274
Tabelle 151: RESET RETRY COUNTER Antwort-APDU im Fehlerfall	275
Tabelle 152: VERIFY, Vergleich eines Benutzergeheimnisses	277
Tabelle 153: VERIFY Antwort-APDU im Erfolgsfall	277
Tabelle 154: VERIFY Antwort-APDU im Fehlerfall	277
Tabelle 155: EXTERNAL AUTHENTICATE ohne Antwortdaten	280
Tabelle 156: MUTUAL AUTHENTICATE mit Antwortdaten.....	281
Tabelle 157: EXTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall.....	281
Tabelle 158: EXTERNAL AUTHENTICATE Antwort-APDU im Fehlerfall.....	281
Tabelle 159: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 1a.....	288
Tabelle 160: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 2a.....	289
Tabelle 161: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 3a.....	290
Tabelle 162: GENERAL AUTHENTICATE PACE Endnutzerkarte, Schritt 4a.....	290
Tabelle 163: GENERAL AUTHENTICATE gegenseitige ELC-Authentisierung, Schritt 1	291
Tabelle 164: GENERAL AUTHENTICATE gegenseitige ELC-Authentisierung, Schritt 2	291
Tabelle 165: GENERAL AUTHENTICATE, asynchrone, sym. Admin., Schritt 1	292
Tabelle 166: GENERAL AUTHENTICATE, asynchrone, sym. Admin., Schritt 2.....	293
Tabelle 167: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 1b.....	293
Tabelle 168: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 2b.....	294
Tabelle 169: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 3b.....	295
Tabelle 170: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 4b.....	295
Tabelle 171: GENERAL AUTHENTICATE PACE Sicherheitsmodul, Schritt 5b.....	296
Tabelle 172: GENERAL AUTHENTICATE, asynchrone, asym. Admin., Schritt 2.....	296
Tabelle 173: GENERAL AUTHENTICATE Antwort-APDU im Erfolgsfall	297
Tabelle 174: GENERAL AUTHENTICATE Antwort-APDU im Fehlerfall	297
Tabelle 175: GET SECURITY STATUS KEY, symmetrischer Schlüssel.....	309
Tabelle 176: GET SECURITY STATUS KEY, Rollenkennung.....	309
Tabelle 177: GET SECURITY STATUS KEY, <i>bitSecurityList</i>	310
Tabelle 178: GET SECURITY STATUS KEY Antwort-APDU im Erfolgsfall	310
Tabelle 179: GET SECURITY STATUS KEY Antwort-APDU im Fehlerfall.....	310
Tabelle 180: INTERNAL AUTHENTICATE	312
Tabelle 181: INTERNAL AUTHENTICATE Antwort-APDU im Erfolgsfall.....	313

Tabelle 182: INTERNAL AUTHENTICATE Antwort-APDU im Fehlerfall	313
Tabelle 183: Tabelle aller PSO Kommando Header	316
Tabelle 184: PSO Compute Cryptographic Checksum, Berechnen eines MAC	317
Tabelle 185: PSO Compute Cryptographic Checksum Antwort-APDU im Erfolgsfall.....	318
Tabelle 186: PSO Compute Cryptographic Checksum Antwort-APDU im Fehlerfall	318
Tabelle 187: PSO Compute Digital Signature, Signieren ohne „message recovery“	320
Tabelle 188: PSO Compute Digital Signature, Signieren „message recovery“	321
Tabelle 189: PSO Compute Digital Signature Antwort-APDU im Erfolgsfall	321
Tabelle 190: PSO Compute Digital Signature Antwort-APDU im Fehlerfall	321
Tabelle 191: PSO Decipher, Entschlüsseln mittels RSA	324
Tabelle 192: PSO Decipher, Entschlüsseln mittels elliptischer Kurven.....	324
Tabelle 193: PSO Decipher, Entschlüsseln mittels symmetrischem Schlüssel.....	325
Tabelle 194: PSO Decipher Antwort-APDU im Erfolgsfall	325
Tabelle 195: PSO Decipher Antwort-APDU im Fehlerfall	325
Tabelle 196: PSO Encipher, Verschlüsseln mittels übergebenem RSA-Schlüssel	329
Tabelle 197: PSO Encipher, Verschlüsseln mittels übergebenem ELC-Schlüssel.....	330
Tabelle 198: PSO Encipher, Verschlüsseln mittels gespeichertem RSA-Schlüssel.....	331
Tabelle 199: PSO Encipher, Verschlüsseln mittels gespeichertem ELC-Schlüssel	331
Tabelle 200: PSO Encipher, Verschlüsseln mittels symmetrischem Schlüssel.....	332
Tabelle 201: PSO Encipher Antwort-APDU im Erfolgsfall	332
Tabelle 202: PSO Encipher Antwort-APDU im Fehlerfall	332
Tabelle 203: PSO Transcipher, Umschlüsseln von Daten mittels RSA-Schlüssel	336
Tabelle 204: PSO Transcipher, Umschlüsseln von Daten von RSA nach ELC.....	337
Tabelle 205: PSO Transcipher, Umschlüsseln von Daten mittels ELC-Schlüssel.....	338
Tabelle 206: PSO Transcipher, Umschlüsseln von Daten von ELC nach RSA.....	340
Tabelle 207: PSO Transcipher Antwort-APDU im Erfolgsfall.....	340
Tabelle 208: PSO Transcipher Antwort-APDU im Fehlerfall.....	340
Tabelle 209: PSO Verify Certificate für RSA-Schlüssel	344
Tabelle 210: Kombination von Kurvenparametern in CV-Zertifikaten	344
Tabelle 211: PSO Verify Certificate für ELC-Schlüssel	345
Tabelle 212: PSO Verify Certificate Antwort-APDU im Erfolgsfall.....	345
Tabelle 213: PSO Verify Certificate Antwort-APDU im Fehlerfall	345
Tabelle 214: PSO Verify Cryptographic Checksum, Prüfen MAC.....	350
Tabelle 215: PSO Verify Cryptographic Checksum Antwort-APDU im Erfolgsfall.....	350
Tabelle 216: PSO Verify Cryptographic Checksum Antwort-APDU im Fehlerfall.....	351
Tabelle 217: PSO Verify Digital Signature mittels übergebenem ELC-Schlüssel.....	353

Tabelle 218: PSO Verify Digital Signature Antwort-APDU im Erfolgsfall.....	353
Tabelle 219: PSO Verify Digital Signature Antwort-APDU im Fehlerfall	353
Tabelle 220: FINGERPRINT über das COS.....	355
Tabelle 221: FINGERPRINT Antwort-APDU im Erfolgsfall.....	355
Tabelle 222: FINGERPRINT Antwort-APDU im Fehlerfall.....	355
Tabelle 223: GAKP, ohne Überschreiben, ohne Referenz, ohne Ausgabe.....	357
Tabelle 224: GAKP, ohne Überschreiben, mit Schlüsselreferenz, ohne Ausgabe	357
Tabelle 225: GAKP, ggf. Überschreiben, ohne Referenz, ohne Ausgabe.....	358
Tabelle 226: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, ohne Ausgabe	358
Tabelle 227: GAKP, Auslesen vorhandener Schlüssel ohne Schlüsselreferenz	359
Tabelle 228: GAKP, Auslesen vorhandener Schlüssel mit Schlüsselreferenz	360
Tabelle 229: GAKP, ohne Überschreiben, ohne Schlüsselreferenz, mit Ausgabe	360
Tabelle 230: GAKP, ohne Überschreiben, mit Schlüsselreferenz, mit Ausgabe	361
Tabelle 231: GAKP, ggf. Überschreiben, ohne Schlüsselreferenz, mit Ausgabe	362
Tabelle 232: GAKP, ggf. Überschreiben, mit Schlüsselreferenz, mit Ausgabe	362
Tabelle 233: GENERATE ASYMMETRIC KEY PAIR, Kommandoparameter im Überblick.....	363
Tabelle 234: GAKP Antwort-APDU im Erfolgsfall	363
Tabelle 235: GAKP Antwort-APDU im Fehlerfall	363
Tabelle 236: GET CHALLENGE für DES oder RSA Authentisierung.....	366
Tabelle 237: GET CHALLENGE für AES oder ELC Authentisierung	366
Tabelle 238: GET CHALLENGE Antwort-APDU im Erfolgsfall.....	366
Tabelle 239: GET CHALLENGE Antwort-APDU im Fehlerfall.....	367
Tabelle 240: GET RANDOM	368
Tabelle 241: GET RANDOM Antwort-APDU im Erfolgsfall	368
Tabelle 242: GET RANDOM Antwort-APDU im Fehlerfall	368
Tabelle 243: LIST PUBLIC KEY mit allen Arten öffentlicher Schlüsselobjekte.....	369
Tabelle 244: LIST PUBLIC KEY Antwort-APDU im Erfolgsfall.....	370
Tabelle 245: LIST PUBLIC KEY Antwort-APDU im Fehlerfall.....	370
Tabelle 246: MANAGE CHANNEL zum Öffnen eines logischen Kanals.....	372
Tabelle 247: MANAGE CHANNEL zum Schließen eines logischen Kanals	373
Tabelle 248: MANAGE CHANNEL zum Zurücksetzen eines logischen Kanals	373
Tabelle 249: MANAGE CHANNEL zum logischen Reset der Applikationsebene	374
Tabelle 250: MANAGE CHANNEL Antwort-APDU im Erfolgsfall.....	374
Tabelle 251: MANAGE CHANNEL Antwort-APDU im Fehlerfall.....	374
Tabelle 252: MSE, Restore Variante.....	377
Tabelle 253: MSE, Selektion symmetrischer INTERNAL AUTHENTICATE-Schlüssel	377

Tabelle 254: MSE, Selektion privater INTERNAL AUTHENTICATE-Schlüssel	378
Tabelle 255: MSE, Selektion symmetrischer EXTERNAL AUTHENTICATE-Schlüssel.....	379
Tabelle 256: MSE, Selektion öffentlicher EXTERNAL AUTHENTICATE-Schlüssel	380
Tabelle 257: MSE, Selektion symmetrischer MUTUAL AUTHENTICATE-Schlüssel.....	381
Tabelle 258: MSE, Selektion sym. Kartenverbindungsobjekt ohne Kurvenangabe.....	382
Tabelle 259: Zuordnung von <i>idDomainParameter</i> zu Domainparameter	382
Tabelle 260: MSE, Selektion symmetrisches Kartenverbindungsobjekt	383
Tabelle 261: MSE, Selektion privater Signaturschlüssel	384
Tabelle 262: MSE, Selektion öffentlicher Zertifikatsprüfchlüssel.....	384
Tabelle 263: MSE, Schlüsselselektion zur Entschlüsselung.....	385
Tabelle 264: MSE, Selektion Verschlüsselungsschlüssel.....	386
Tabelle 265: MSE Antwort-APDU im Erfolgsfall	386
Tabelle 266: MSE Antwort-APDU im Fehlerfall	386
Tabelle 267: Generische AlgorithmIdentifier für Authentisierungszwecke	401
Tabelle 268: Konkrete AlgorithmIdentifier für Authentisierungszwecke	401
Tabelle 269: AlgorithmIdentifier für Ver- und Entschlüsselung	402
Tabelle 270: AlgorithmIdentifier für Integrität und Authentizität	402
Tabelle 271: Object Identifier, alphabetisch sortiert (informativ)	402
Tabelle 272: Trailer → Fehlernamen.....	404
Tabelle 273: Bedeutung PPS1 gemäß [ISO/IEC 7816-3#Table 7 und 8]	408
Tabelle 274: Gesamtbewertung für das Basisbetriebssystem	412
Tabelle 275: Gesamtbewertung für Option_kontaktlose_Schnittstelle.....	414
Tabelle 276: Gesamtbewertung für Option_Kryptobox.....	414
Tabelle 277: Gesamtbewertung für Option_logische_Kanäle.....	415
Tabelle 278: Gesamtbewertung je nach Kombination der Optionen.....	415
Tabelle 279: Character Guard Time (CGT) gemäß [ISO/IEC 7816-3#11.2]	416
Tabelle 280: ATR Codierung.....	457
Tabelle 281: Attribute / MF	458
Tabelle 282: Attribute / MF / EF.ATR	458
Tabelle 283: Attribute / MF / EF.DIR	459
Tabelle 284: Attribute / MF / EF.GDO	460
Tabelle 285: Attribute / MF / DF.Auth.....	460
Tabelle 286: Attribute / MF / DF.Auth / PrK.Auth_ELC256	461
Tabelle 287: Attribute / MF / DF.Auth / PrK.Auth_ELC384	461
Tabelle 288: Attribute / MF / DF.Auth / PrK.Auth_ELC512	462
Tabelle 289: Attribute / MF / DF.Auth / PuK.RCA_ELC256	462

Tabelle 290: Attribute / MF / DF.Auth / PuK.RCA_ELC384	463
Tabelle 291: Attribute / MF / DF.Auth / PuK.RCA_ELC512	463
Tabelle 292: Attribute / MF / DF.Auth / SK.AES128	464
Tabelle 293: Attribute / MF / DF.Auth / SK.AES192	464
Tabelle 294: Attribute / MF / DF.Auth / SK.AES256	464
Tabelle 295: Attribute / MF / DF.Auth / TC.AES128	465
Tabelle 296: Attribute / MF / DF.Auth / TC.AES192	465
Tabelle 297: Attribute / MF / DF.Auth / TC.AES256	465
Tabelle 298: Attribute / MF / DF.IAS	466
Tabelle 299: Attribute / MF / DF.IAS / PrK.X509_ELC256	466
Tabelle 300: Attribute / MF / DF.IAS / PrK.X509_ELC384	466
Tabelle 301: Attribute / MF / DF.IAS / PrK.X509_ELC512	467
Tabelle 302: Attribute / MF / DF.IAS / PrK.X509_RSA2048	468
Tabelle 303: Attribute / MF / DF.IAS / PrK.X509_RSA3072	469
Tabelle 304: Attribute / MF / DF.LCS	469
Tabelle 305: Attribute / MF / DF.LCS / CAN_256	470
Tabelle 306: Attribute / MF / DF.LCS / EF.LCS	470
Tabelle 307: Attribute / MF / DF.LCS / PIN.LCS	471
Tabelle 308: Attribute / MF / DF.LCS / PrK.LCS	471
Tabelle 309: Attribute / MF / DF.LCS / PuK.LCS	472
Tabelle 310: Attribute / MF / DF.LCS / SK.LCS	472
Tabelle 311: Attribute / MF / DF.SelectEF	472
Tabelle 312: Attribute / MF / DF.SelectEF / EF.xx	473
Tabelle 313: Attribute / MF / DF.strukturiert	473
Tabelle 314: Attribute / MF / DF.strukturiert / EF.strukturiert	473
Tabelle 315: Attribute / MF / DF.transparent	474
Tabelle 316: Attribute / MF / DF.transparent / EF.transparent	474

E.5 Referenzierte Dokumente

E.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 veröf-

fentlichten 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

E.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, <i>The Elliptic Curve Digital Signature Algorithm (ECDSA)</i> , 2005
[ANSI X9.63]	Public Key Cryptography for the Financial Services Industry, <i>Key Agreement and Key Transport Using Elliptic Curve Cryptography</i> , 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

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[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 (2 nd edition), Identification cards — Integrated circuit cards with contacts — Part 3: Electrical interface and transmission protocols
[ISO/IEC 7816-4]	ISO/IEC 7816-4: 2013 (3 rd edition) Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange
[ISO/IEC 7816-5]	ISO/IEC 7816-5: 2004 (2 nd edition) Identification cards — Integrated circuit cards — Part 5: Registration of application providers
[ISO/IEC 7816-8]	ISO/IEC 7816-8: 2016 (3 rd edition) Identification cards — Integrated circuit cards — Part 8: Commands and mechanisms for security operations
[ISO/IEC 7816-9]	ISO/IEC 7816-9: 2004 (2 nd edition) Identification cards — Integrated circuit cards — Part 9: Commands for card management
[ISO/IEC 7816-12]	ISO/IEC 7816-12: 2005-10-04 (1 st 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 (1 st 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 (3 rd 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 (3 rd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 1: Physical characteristics
[ISO/IEC 14443-2]	ISO/IEC 14443-2: 2016-07 (3 rd edition)

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
	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 (3 rd edition) Identification cards — Contactless integrated circuit cards — Proximity cards — Part 3: Initialization and anticollision
[ISO/IEC 14443-4]	ISO/IEC 14443-4: 2016-06 (3 rd 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, <i>Recommendation for Block, Cipher Modes of Operation, Methods and Techniques</i> , 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 – Part 2 – Extended Access Control Version 2 (EACv2), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI), Version 2.10, 2012-03-20
[BSI-TR-03110-3]	Technical Guideline TR-03110-3, Advanced Security Mechanisms for Machine Readable Travel Documents – Part 3 – Common Specifications Version 2.11, 2013-07-12
[BSI-TR-03111]	Technical Guideline TR-03111, Elliptic Curve Cryptography, Version 2.0, 2012-06-28
[BSI-TR-03116-1]	Technische Richtlinie BSI TR-03116-1, Kryptographische Vorgaben für Projekte der Bundesregierung, Version: 3.19, Datum: 3.12.2015, Status: Veröffentlichung, Fassung: Dezember 2015

Anhang F – Asynchrone, gesicherte APDU-Sequenzen (informativ)

F.1 Einleitung

Gemäß einer Anforderung an das COS der Generation 2 soll es einem Kartenmanagement (CMS) möglich sein nicht nur synchron, sondern auch asynchron mit Karten der Generation 2 zu kommunizieren. Dieses Papier beleuchtet die diesbezügliche Anforderungslage und zeigt eine technische Umsetzungsmöglichkeit.

F.2 CMS-Kommunikationsmuster

Hier werden das synchrone und das asynchrone Kommunikationsmuster zwischen einem Kartenmanagement und einer Smartcard aus technischer Sicht betrachtet.

F.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.

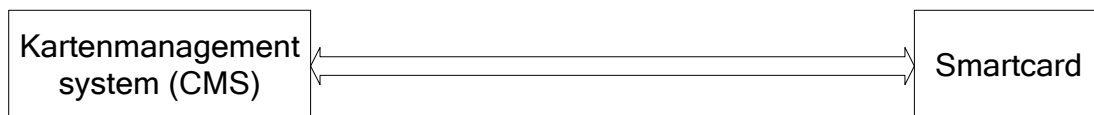


Abbildung 13 - Synchrone Kommunikation zw. CMS und Smartcard

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.

F.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.

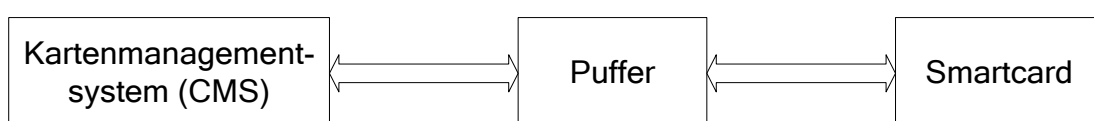


Abbildung 14 - Asynchrone Kommunikation zw. CMS und Smartcard

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.

F.3 Anforderungen an die asynchrone Kommunikation

Aus sicherheitstechnischer Sicht werden an die asynchrone Kommunikation folgende Anforderungen gestellt:

- 4) 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.
- 5) 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.
- 6) Authentizität: Das CMS MUSS sich zweifelsfrei als Quelle eines Szenarios und jeder einzelnen Kommando-APDU innerhalb eines Szenarios identifizieren lassen.
- 7) 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
- 8) Vertraulichkeit: Es MUSS möglich sein, die Vertraulichkeit von Informationen zu gewährleisten, die in einem Szenario transportiert werden.
- 9) 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 sollen. Eine Aktualisierung von Anwendungsdaten ist nur dann sinnvoll, wenn die Anwendung geladen wurde.

- 10) Resistenz gegen Replay Attacks: Es ist auszuschließen, dass ein Szenario ein zweites Mal erfolgreich durchgeführt wird.
- 11) 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.

F.4 Lösungskonzept

Zur Umsetzung der in F.3 vorgeschlagenen Anforderungen wird folgendes Konzept vorgeschlagen:

- 12) 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).
- 13) NumberScenario: Jedem Szenario wird vom CMS eine natürliche Zahl zugeordnet. Die Zuordnung zwischen Szenario und NumberScenario MUSS bijektiv (d.h. umkehrbar eindeutig) sein.
- 14) 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).
- 15) Die Smartcard speichert NumberScenario eines Szenarios. Szenarien MÜSSEN so an die Smartcard geschickt werden, dass ihr NumberScenario streng monoton steigt (Anforderungen Resistent gegen Replay Attacks). 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.

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.

F.5 Kryptographie

Zur Erfüllung der in F.3 genannten kryptographischen Anforderungen werden symmetrische und asymmetrische Authentisierungsschlüssel verwendet.

F.5.1 CMS Aktivitäten

Auf Seiten des CMS ist der Ablauf wie folgt:

16) 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 Kapitel 13 gesichert.

17) 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:
 $\text{cipherText} = \text{AES_CBC_ENC}(\text{SK.CMS_ENC}, \text{ICV} = '00...00', \text{Seed})$.
 - ii) asymmetrische Administration:
 - (1) $(PO_A, C, T) = \text{ELC_ENC}(\text{Seed}, \text{PuK.SmartCard.AUT}, \text{PuK.CMS.dP})$
 - (2) Setze $\text{oidDO} = '06-L_{06}-\text{affectedObject.domainParameter.OID}'$.
 - (3) Setze $\text{keyDO} = '7F49-L_{7F49}-(86-L_{86}-PO_A)'$.
 - (4) Setze $\text{cipherDO} = '86-L_{86}-(02 \parallel C)'$.
 - (5) Setze $\text{macDO} = '8E-L_{8E}-T'$.
 - (6) Setze $\text{cipherText} = 'A6-L_{A6}-(oidDO \parallel keyDO \parallel cipherDO \parallel macDO)'$.
- c) Es wird eine Nachricht M wie folgt zusammengestellt:
 $M = \text{NumberPrecondition} \parallel \text{NumberScenario} \parallel \text{cipherText}$.
- d) Für die Nachricht M werden Authentisierungsdaten berechnet:
 - i) symmetrische Administration: Es wird ein CMAC für M berechnet:
 $\text{CMAC} = \text{CalculateCMAC_IsoPadding}(\text{SK.CMS_MAC}, M)$.
 - ii) asymmetrische Administration: Es wird eine Signatur für M berechnet:

- (1) Falls $\text{PrK.CMS.dP} = \text{brainpool256} \Rightarrow \text{hash} = \text{SHA_256}(M)$.
 - (2) Falls $\text{PrK.CMS.dP} = \text{brainpool384} \Rightarrow \text{hash} = \text{SHA_384}(M)$.
 - (3) Falls $\text{PrK.CMS.dP} = \text{brainpool512} \Rightarrow \text{hash} = \text{SHA_512}(M)$.
 - (4) $(R, S) = \text{ELC_SIG}(\text{PrK.CMS}, \text{hash})$
 - (5) $\text{SIG} = R \parallel S$.
- e) Eine Authentisierungssequenz mit M und CMAC bzw. SIG wird dem Szenario aus Schritt 16)c) vorangestellt.

F.5.2 Puffer-Aktivitäten

Auf Seiten des Puffers ist der Ablauf im Gutfall wie folgt:

- 18) Der Puffer ermittelt von einer Smartcard die Werte ICCSN und NumberScenario.
- 19) Der Puffer wählt ein passendes Szenario aus.
- 20) Der Puffer schickt das ausgewählte Szenario an die Smartcard.
- 21) Der Puffer schickt die gesicherten Antwort-APDU zusammen mit ICCSN und NumberScenario als Quittung an das CMS.

F.5.3 Smartcard-Aktivitäten

Auf Seiten der Smartcard ist der Ablauf wie folgt:

- 22) Schlüsselauswahl: Zu Beginn des Szenarios werden ein passendes Schlüsselobjekt und ein passender Algorithmus ausgewählt (etwa durch ein MSE-Set-Kommando).
- 23) 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) Falls $\text{NumberPrecondition}$ in M größer als $\text{SK.CMS} \rightarrow \text{numberScenario}$ bzw. $\text{PrK.SmartCard.AUT} \rightarrow \text{numberScenario}$ ist, führt dies zum Kommandoabbruch.
 - ii) Falls NumberScenario in M kleiner oder gleich $\text{SK.CMS} \rightarrow \text{numberScenario}$ bzw. $\text{PrK.SmartCard.AUT} \rightarrow \text{numberScenario}$ ist, führt dies zum Kommandoabbruch.
 - c) $\text{SK.CMS} \rightarrow \text{numberScenario}$ bzw. $\text{PrK.SmartCard.AUT} \rightarrow \text{numberScenario}$ 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 SendSequenceCounter verwendet.

- 24) Kartenmanagementkommandos: Die übrigen, Secure Messaging gesicherten Kommando-APDU werden an die Smartcard geschickt. Secure-Messaging-Fehler führen (wie üblich) zum Kommandoabbruch.

F.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) MSE Set P1P2='81A4', keyReference=SK.CMS, 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.

Anhang G – CV-Zertifikate für RSA-Schlüssel, Option_RSA_CVC

Der Inhalt dieses Anhangs wurde nach [gemSpec_PKI] verschoben.

Anhang H – CV-Zertifikate für ELC-Schlüssel (normativ)

Der Inhalt dieses Anhangs wurde nach [gemSpec_PKI] verschoben.

Anhang I – 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 genau so 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.

I.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 (**End-Entity**), 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.

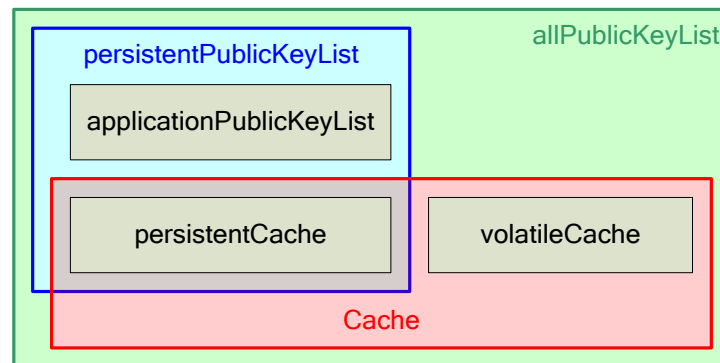


Abbildung 15: Zusammenhang Schlüsselspeicher

I.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 importierte Schlüssel dem Cache hinzugefügt. Es wird davon ausgegangen, dass der Cache nur eine begrenzte Kapazität hat. Falls die Menge an importierten Schlüsseln die Kapazität des Cache übersteigt, werden „unwichtige“ Einträge aus dem Cache entfernt um Platz für den importierten Schlüssel zu schaffen.