# Inhaltsverzeichnis

# 1 Änderungsbedarf

In Vorbereitung auf ein Entfernen von RSA-2048 aus der TI wird die Spezifikation des Basis-Consumer so überarbeitet, dass der Basis-Consumer für erzeugende Kryptooperationen immer ECC-Zertifikate verwendet

Lesende Kryptooperationen werden nicht geändert und bereits existierendes RSA-basiertes Material kann weiterhin vom Basis-Consumer verwendet werden.

# 2 Änderungen in gemSpec_CM_KOMLE

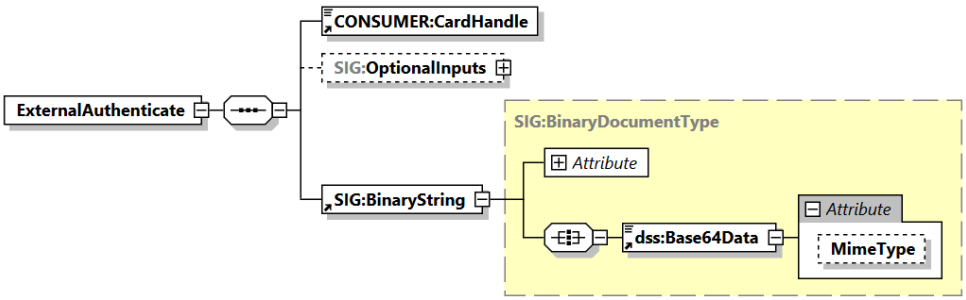
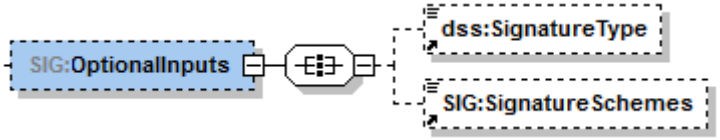
Siehe separaten Änderungseintrag C_11832

# 3 Änderungen in gemSpec_Basis_KTR_Consumer

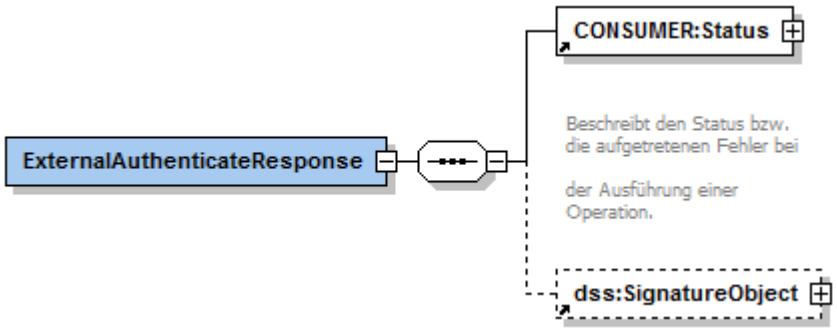
A_17578-03 wird durch A_17578-04 ersetzt:

**A_17578-04 - Basis- und KTR-Consumer, Operation ExternalAuthenticate**
Der Signaturdienst des Basis- und KTR-Consumer MUSS an der Client-Systemschnittstelle die Operation `ExternalAuthenticate` wie in Tabelle Tab_Operation_ExternalAuthenticate beschrieben anbieten.

**Tabelle 1: Tab_Operation_ExternalAuthenticate**

<table>
<tr><th>Name</th><td colspan="2">ExternalAuthenticate</td></tr>
<tr><th>Beschreibung</th><td colspan="2">Diese Operation versieht einen Binärstring der maximalen Länge ~~512~~256 Bit mit einer nicht-qualifizierten elektronischen Signatur (nonQES). Dazu wird das Signaturverfahren ~~PKCS#1 oder~~ ECDSA verwendet.</td></tr>
<tr><th>Aufrufparameter</th><td colspan="2"> 
</td></tr>
<tr><td></td><th>Name</th><th>Beschreibung</th></tr>
<tr><td></td><td>CONSUMER:<br>CardHandle</td><td>Identifiziert die zu verwendende Signaturquelle.<br>Die Operation unterstützt nur Identitäten gem. Hinweis unter Tabelle 7 Tab_Personalisierung_HSM.</td></tr>
<tr><td></td><td>~~SIG:<br>Optional<br>Inputs~~</td><td>~~Enthält optionale Eingangsparameter:~~<br> 
</td></tr>
</table>

| Name | ExternalAuthenticate | |
|---|---|---|
| | SIG: BinaryString | Dieses Element enthält im Kindelement dss:Base64Data den zu signierenden Binärstring.<br>Das XML Attribut SIG:BinaryString/dss:Base64Data/@MimeType MUSS den Wert "application/octet-stream" haben.<br>Die maximale Länge des Binärstrings beträgt ~~512~~256 Bit entsprechend der maximal zu erwartenden Hash-Größe.<br>~~Aus der Länge des Binärstrings wird auf das verwendete Hashverfahren geschlossen. Es werden folgende Längen unterstützt:~~<br>• ~~256 Bit: SHA-256 (OID 2.16.840.1.101.3.4.2.1)~~<br>• ~~384 Bit: SHA-384 (OID 2.16.840.1.101.3.4.2.2)~~<br>• ~~512 Bit: SHA-512 (OID 2.16.840.1.101.3.4.2.3)~~<br>~~Im Falle des Signaturverfahrens RSASSA-PKCS1-v1_5 werden SHA-256, SHA-384 und SHA-512 unterstützt.~~<br>~~Im Falle des Signaturverfahrens RSASSA-PSS wird SHA-256 unterstützt.~~<br>~~Im Falle des Signaturverfahrens ECDSA wird SHA-256 unterstützt.~~<br>Als Signaturverfahren wird ausschließlich ECDSA unterstützt.<br>Für die Signaturerstellung gilt:<br>• ~~Im Falle des Signaturverfahrens RSASSA-PKCS1-v1_5 wird die Ausführung der Methode EMSA-PKCS1-v1_5-ENCODE nach [RFC3447], Abschnitt 9.2, mit Schritt 2, Erstellung des DigestInfo-Datenfeldes begonnen.~~<br>• ~~Im Falle des Signaturverfahrens RSASSA-PSS wird die Ausführung der Methode EMSA-PSS-ENCODE nach [RFC3447], Abschnitt 9.1.1, mit Schritt 3 begonnen.~~<br>• ~~Im Falle des Signaturverfahrens ECDSA~~ Es erfolgt die Signaturerstellung gemäß [BSI-TR-03111]#4.2.1. Als Eingangsparameter wird der Hash vom Aufrufer in SIG: BinaryString übergeben. |

| Name | ExternalAuthenticate | |
|---|---|---|
| | ~~dss:SignatureType~~ | ~~Durch dieses in [OASIS-DSS] (Abschnitt 3.5.1) beschriebene Element wird der Typ der zu erzeugenden Signatur bestimmt. Als Signaturtyp wird unterstützt :~~<br>• ~~**PKCS#1-Signatur**~~<br>~~Durch Übergabe der URI urn:ietf:rfc:3447 wird eine PKCS#1 (Version 2.1) Signatur gemäß [RFC3447] erzeugt, die als dss:Base64Signature mit der oben genannten URI zurückgeliefert wird.~~<br>• ~~**ECDSA-Signatur**~~<br>~~Durch Übergabe der URI urn:bsi:tr:03111:ecdsa wird eine ECDSA-Signatur gemäß [BSI-TR-03111]#4.2.1 erzeugt, die als dss:Base64Signature mit der oben genannten URI zurückgeliefert wird.~~<br>~~Andere SignatureType-Angaben führen zu Fehlermeldung 4111 (Ungültiger Signaturtyp oder Signaturvariante).~~<br>~~Fehlt dieses Element, so wird ebenfalls der Signaturtyp ECDSA-Signatur verwendet.~~ |
| | ~~SIG:SignatureSchemes~~ | ~~Durch dieses Element wird für PKCS#1-Signaturen zwischen den folgenden SignatureScheme-Optionen unterschieden:~~<br>• ~~RSASSA-PSS~~<br>• ~~RSASSA-PKCS1-v1_5~~<br>~~Fehlt dieses Element, so wird als Default-SignatureScheme RSASSA-PSS gewählt.~~ |
| **Rückgabe** |  | |
| | CONSUMER:Status | Enthält den Status der ausgeführten Operation. |

| Name | ExternalAuthenticate | |
|---|---|---|
| | dss:SignatureObject | Enthält im Erfolgsfall die erzeugte Signatur in Form eines dss:SignatureObject-Elements gemäß [OASIS-DSS] (Abschnitt 3.2).<br>Der Signaturwert wird im XML-Element dss:SignatureObject/dss:Base64Signature übergeben. Das XML-Attribut dss:SignatureObject/dss:Base64Signature/@Type kennzeichnet durch den Wert:<br>• ~~urn:ietf:rfc:3447 den Signatur-Typ PKCS#1~~<br>~~bzw.~~<br>• urn:bsi:tr:03111:ecdsa den Signatur-Typ ECDSA.<br>Die XML-Elemente<br>dss:SignatureObject/ds:Signature<br>dss:SignatureObject/dss:Timestamp<br>dss:SignatureObject/dss:SignaturePtr<br>dss:SignatureObject/dss:Other<br>werden nicht verwendet. |
| **Vorbedingungen** | Keine | |
| **Nachbedingungen** | Keine | |

Der Ablauf der Operation ExternalAuthenticate ist in Tabelle Tab_Ablauf_ExternalAuthenticate beschrieben:

**Tabelle 2: Tab_Ablauf_ExternalAuthenticate**

| Nr. | Aufruf Technischer Use Case oder Interne Operation | Beschreibung |
|---|---|---|
| 1. | checkArguments | Alle übergebenen Parameterwerte werden auf Konsistenz und Gültigkeit überprüft. Treten hierbei Fehler auf, so bricht die Operation mit Fehler 4000 ab. |

| Nr. | Aufruf Technischer Use Case oder Interne Operation | Beschreibung |
|---|---|---|
| 2. | Signiere | Das Signieren erfolgt durch Aufruf von PL_TUC_SIGN_HASH_nonQES {<br>IDENTIFIKATOR = CardHandle;<br>SIGNATURVERFAHREN = ~~SIG:SignatureSchemes~~ECDSA mit SHA-256;<br>HASHWERT = SIG:BinaryString;<br>}<br><br>Tritt hierbei ein Fehler auf, so bricht die Operation mit Fehler 4123 ab. |

**Tabelle 3: Tab_Fehler_ExternalAuthenticate**

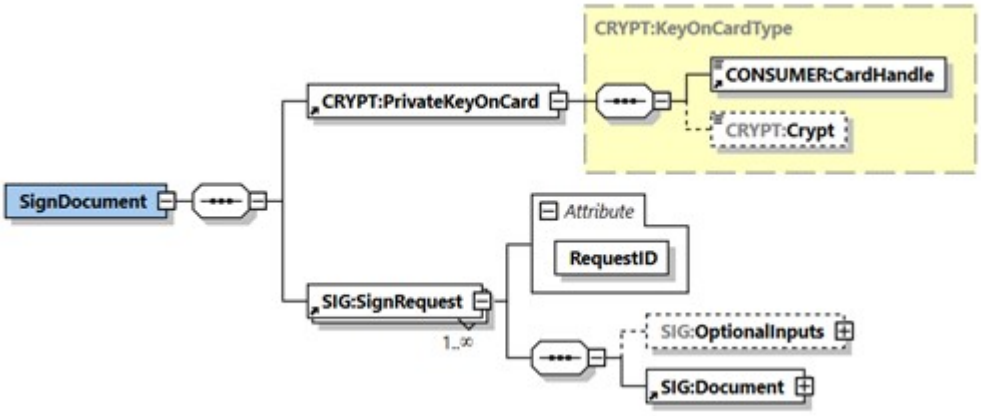
| Fehlercode | ErrorType | Severity | Fehlertext |
|---|---|---|---|
| Neben den Fehlercodes der aufgerufenen TUCs können folgende weitere Fehlercodes auftreten: | | | |
| 4000 | Technical | Error | Syntaxfehler |
| 4123 | Technical | Error | Fehler bei Signaturerstellung |
| ~~4111~~ | ~~Technical~~ | ~~Error~~ | ~~Ungültiger Signaturtyp oder Signaturvariante~~ |

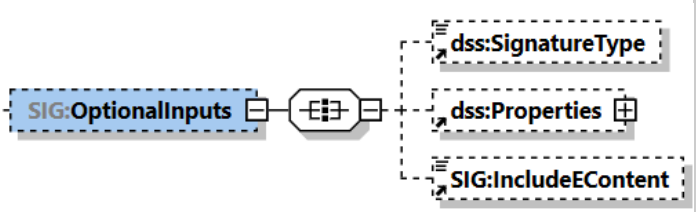
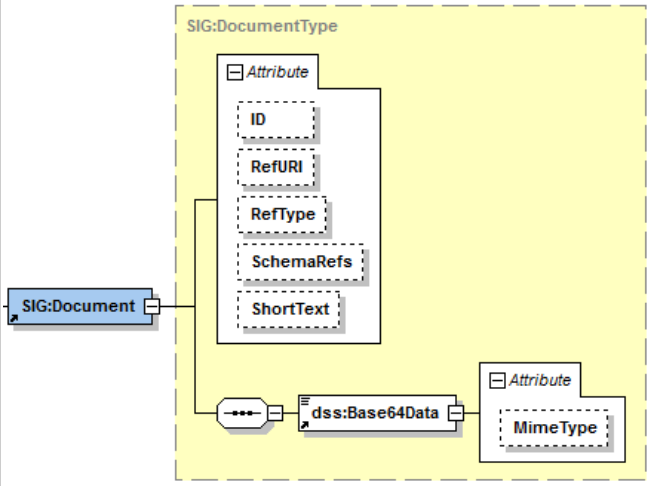
Der zulässige private Schlüssel für die Nutzung durch die Operation ExternalAuthenticate ist für SM-B PrK.HCI.AUT in DF.ESIGN.
**[**<=, Basis-Consumer, Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA**]**

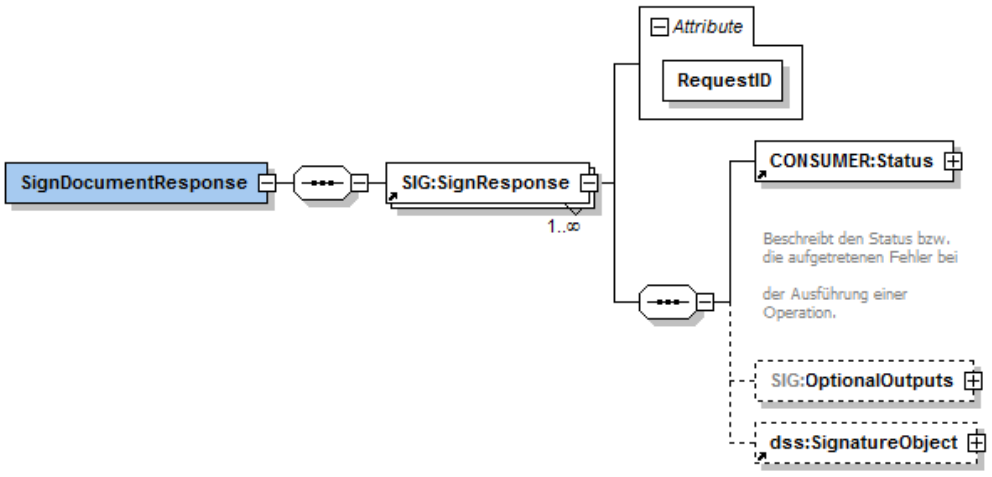
A_17525-03 wird durch A_17525-04 ersetzt:

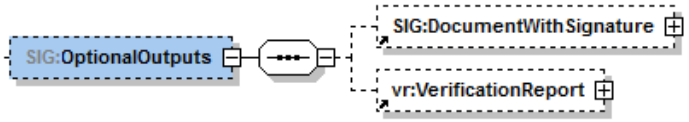
**A_17525-04 - Basis- und KTR-Consumer, Operation SignDocument**
Der Signaturdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle eine an [OASIS-DSS] angelehnte Operation `SignDocument` wie in Tabelle Tab_Operation_SignDocument beschrieben anbieten.

**Tabelle 4: Tab_Operation_SignDocument**

| **Name** | `SignDocument` | |
|---|---|---|
| **Beschreibung** | Diese Operation lehnt sich an [OASIS-DSS] an. Sie enthält voneinander unabhängige SignRequests. Jeder SignRequest erzeugt eine Signatur für ein Dokument.<br>Zur Signaturerzeugung werden Schlüssel und Zertifikate eines HSM benutzt. Es wird ausschließlich der Signaturtyp "CMS-Signatur" gemäß [RFC 5652] (URIurn:ietf:rfc:5652) und das Profil CAdES-BES gemäß[CAdES] verwendet. | |
| **Aufruf-parameter** |  | |
| | `PrivateKeyOnCard` | Identifiziert die zu verwendende Karte mit dem (privaten) Schlüssel. |
| | `CardHandle` | Identifiziert die zu verwendende Signaturkarte. |
| | `Crypt` | Der Parameter wird nicht ausgewertet - Optional;<br>Es muss ausschließlich ECC über die KeyReference PrK.HCI.OSIG.E256verwendet werden.<br>~~Dieser Parameter steuert die Auswahl der Zertifikate und Schlüssel für die Signaturerstellung. Die Werte sind in der Tabelle Tab_Zertifikate_für_Sign/VerifyDocument vorgegeben.~~<br>~~(Default-Wert ist RSA)~~ |
| | `SIG:SignRequest` | Ein `SignRequest` kapselt den Signaturauftrag für ein Dokument.<br>Das verpflichtende XML-Attribut `RequestID` identifiziert einen `SignRequest` innerhalb eines Stapels von `SignRequests` eindeutig. Es dient der Zuordnung der `SignResponse` zum jeweiligen `SignRequest`. |

| Name | SignDocument | |
|---|---|---|
| | SIG:OptionalInputs | Enthält optionale Eingangsparameter (angelehnt an dss:OptionalInputs gemäß [OASIS-DSS] Section 2.7):<br>  |
| | SIG:Document |  <br>Dieses an das dss:Document Element aus [OASIS-DSS] Section 2.4.2 angelehnte Element enthält das zu signierende Dokument in dss:Base64Data. |
| | dss:SignatureType | Durch dieses in [OASIS-DSS] (Abschnitt 3.5.1) beschriebene Element kann der generelle Typ der zu erzeugenden Signaturen angegeben werden. Es muss der Signaturtyp CMS-Signatur (URI urn:ietf:rfc:5652) unterstützt werden.<br><br>Fehlt dieses Element, so muss der Signaturtyp CMS-Signatur (URI urn:ietf:rfc:5652) implizit verwendet werden. |

| Name | SignDocument | |
|---|---|---|
| | dss:Properties | Durch dieses in [OASIS-DSS] (Abschnitt 3.5.5) definierte Element können zusätzliche signierte und unsignierte Eigenschaften (Properties) bzw. Attribute in die Signatur eingefügt werden<br>.<br>Es dürfen genau die folgenden Attribute<br>./SignedProperties/Property/Value/CMSAttribute<br>und<br>./UnsignedProperties/Property/Value/CMSAttribute<br>enthalten sein.<br>Ein solches XML-Element CMSAttribute muss ein vollständiges, base64/DER-kodiertes ASN.1-Attribute enthalten, definiert in [CMS#5.3.SignerInfo Type]. Es muss bei der Erstellung des CMS-Containers unverändert unter SignedAttributes bzw. UnsignedAttributes aufgenommen werden. |
| | SIG:IncludeEContent | Durch dieses in [OASIS-DSS] (Abschnitt 3.5.7), definierte Element kann bei einer CMS-basierten Signatur das Einfügen des signierten Dokumentes in die Signatur angefordert werden.<br>Fehlt dieses Element oder ist der Wert = "'false', wird die Signaturvariante "detached" verwendet, ansonsten "enveloping". |
| **Rückgabe** |  | |
| | SIG:SignResponse | Eine SignResponse kapselt den ausgeführten Signaturauftrag pro Dokument. Die Zuordnung zwischen SignRequest und SignResponse erfolgt über die RequestID. |
| | CONSUMER:Status | Enthält den Status der ausgeführten Operation pro SignRequest. |

| Name | SignDocument | |
|---|---|---|
| | SIG:OptionalOutputs | Enthält optionale Ausgangsparameter.<br>Dieses Element wird durch den Basis- und KTR-Consumer nicht befüllt.<br>  |
| | SIG:DocumentWithSignature | Dieses Element wird durch den Basis- und KTR-Consumer nicht befüllt. |
| | vr:VerificationReport | Dieses Element wird durch den Basis- und KTR-Consumer nicht befüllt. |
| | dss:SignatureObject | Enthält im Erfolgsfall die erzeugte Signatur in Form eines dss:SignatureObject-Elements gemäß [OASIS-DSS] (Abschnitt 3.2).<br>Der Signaturwert wird im XML-Element dss:SignatureObject/dss:Base64Signature übergeben.<br>Der Signatur-Typ (CMS Signatur) in dss:SignatureObject/dss:Base64Signature/@Type<br><br>Die XML-Elemente<br>dss:SignatureObject/ds:Signature<br>dss:SignatureObject/dss:Timestamp<br>dss:SignatureObject/dss:SignaturePtr<br>dss:SignatureObject/dss:Other<br>werden nicht verwendet. |
| **Vorbedingungen** | Keine | |
| **Nachbedingungen** | Keine | |

Das Signieren erfolgt durch Aufruf von PL_TUC_SIGN_DOCUMENT_nonQES {
  IDENTIFIKATOR = PrivateKeyOnCard;
  DOKUMENT = SIG:Document;
  DOKUMENTTYPE = dss:SignatureType;
}

~~Die folgende Tabelle führt die zulässigen Zertifikate und Schlüssel für die nonQES auf:~~

~~**Tabelle 5: Tab_Zertifikate_für_Sign/VerifyDocument(nonQeS)**~~

| ~~**Karte**~~ | ~~Crypt (Wert)~~ | ~~**KeyReference (Verify)**~~ | ~~**KeyReference (Sign)**~~ |
|---|---|---|---|
| | | ~~in DF.ESIGN~~ | ~~in DF.ESIGN~~ |

| ~~Karte~~ | ~~Crypt (Wert)~~ | ~~KeyReference (Verify)~~ | ~~KeyReference (Sign)~~ |
|---|---|---|---|
| ~~SM-B (KTR/Org) (HSM)~~ | ~~RSA~~ | ~~EF.C.HCI.OSIG.R2048~~ | ~~PrK.HCI.OSIG.R2048~~ |
| | ~~ECC~~ | ~~EF.C.HCI.OSIG.E256~~ | ~~PrK.HCI.OSIG.E256~~ |
| | ~~RSA_ECC~~ | ~~EF.C.HCI.OSIG.R2048~~ ~~EF.C.HCI.OSIG.E256~~ | ~~PrK.HCI.OSIG.R2048~~ ~~PrK.HCI.OSIG.E256~~ |

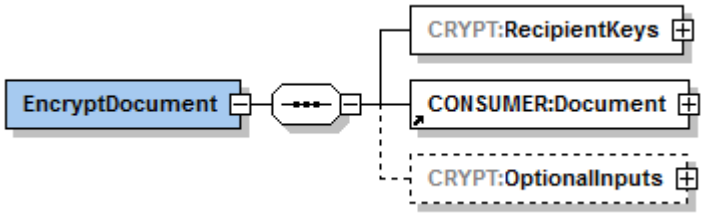
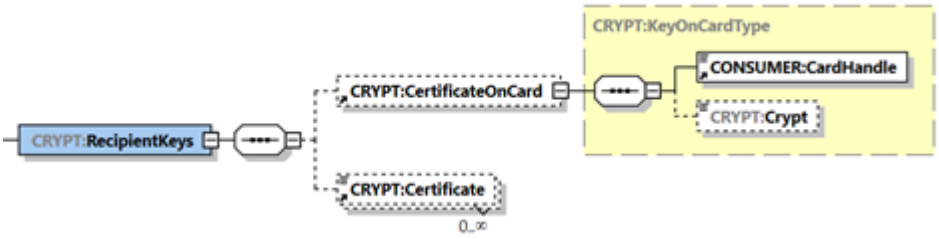
**[**<=, Basis-Consumer, Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA**]**

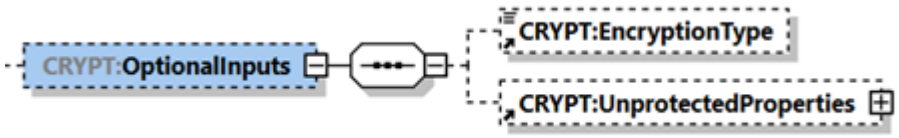
A_17510-04 wird durch A_17510-05 ersetzt:

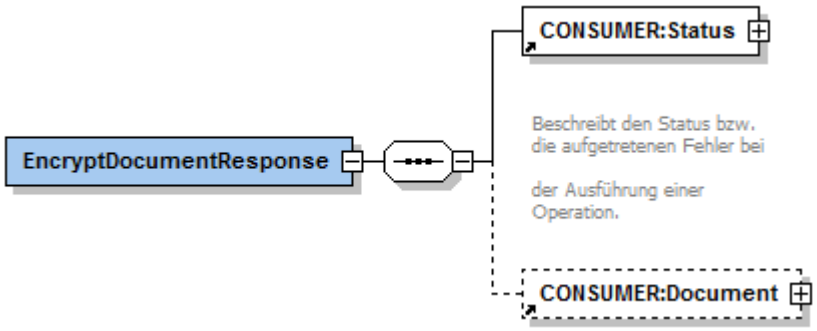
**A_17510-05 - Basis- und KTR-Consumer, Operation EncryptDocument**
Der Verschlüsselungsdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle eine Operation EncryptDocument anbieten.

**Tabelle 6: Tab_Operation_EncryptDocument**

| **Name** | `EncryptDocument` |
|---|---|
| **Beschreibung** | Diese Operation verschlüsselt ein übergebenes Dokument hybrid. Der Dokumententyp XML wird gesondert behandelt. Alle anderen Dokumententypen nutzen die binäre Verschlüsselung.<br>Für die hybride Verschlüsselung wird ein asymmetrischer Schlüssel aus einem X.509v3-Zertifikat genutzt. Dieses Zertifikat wird als Parameter übergeben oder auf dem HSM referenziert. Pro Operationsaufruf können mehrere Hybridschlüssel erzeugt werden. Durch das Zertifikat wird festgelegt, ob RSA oder ECC basierte Hybridschlüssel erzeugt werden.<br>Bei Angabe der Zertifikate über `CertificateOnCard` (Referenz auf HSM) wird ausschließlich das Verschlüsselungsverfahren ECC über die KeyReference EF.C.HCI.ENC.E256 verwendet. ~~das Verschlüsselungsverfahren durch die Angabe in `Crypt` bestimmt. Es können Hybridschlüssel für RSA oder ECC oder beide Verfahren erzeugt werden.~~<br>Für alle Dokumententypen wird immer das gesamte Dokument verschlüsselt. |

| Name | EncryptDocument | |
|---|---|---|
| Aufrufparameter |  | |
| |  | |
| | RecipientKeys | Identifiziert die Empfänger der zu verschlüsselnden Nachricht über X.509-Zertifikate (öffentliche Schlüssel). Quelle für die Zertifikate kann eine Karte sein, die per CertificateOnCard-Element referenziert wird, oder der Aufrufer, der X.509-Zertifikate im Certificate-Element übergibt. |
| | CardHandle | Identifiziert die zu verwendende Karte mit dem (öffentlichen) Schlüssel.<br>Ist das Element nicht vorhanden, so werden nur Zertifikate per Element Certificate übergeben. |
| | Crypt | Der Parameter wird nicht ausgewertet - Optional; ~~Der Wert dieses Parameters ist in Tabelle Tab_KeyReference_für_Encrypt/Decrypt spezifiziert und gibt den Typ von Zertifikaten und dadurch das Verfahren für die Erzeugung der Hybridschlüssel vor. (Default-Wert ist RSA)~~ |
| | Certificate | Certificate ist ein Base64-kodiertes XML-Element, in dem das Zertifikat, das den asymmetrischen Schlüssel enthält (öffentlicher Schlüssel), DER-kodiert übergeben wird.<br>Es kann eine Liste von Zertifikaten übergeben werden.<br>Dieses Element kann leer sein, wenn ausschließlich Zertifikate verwendet werden sollen, die über CertificateOnCard angegeben werden. |

| Name | EncryptDocument | |
|---|---|---|
| | CONSUMER:Document | Dieses entsprechend [OASIS-DSS] Section 2.4.2 spezifizierte Element enthält das zu verschlüsselnde Dokument, wobei das Kindelement dss:Base64Data oder CONSUMER:Base64XML verwendet wird.<br>Das zugeordnete Verschlüsselungsverfahren ist<br>• XMLEnc: „http://www.w3.org/TR/xmlenc-core/" für CONSUMER:Base64XML<br>• CMS: „urn:ietf:rfc:5652" für dss:Base64Data |
| |  | |
| | CRYPT:OptionalInputs | Enthält die optionalen Parameter CRYPT:UnprotectedProperties und CRYPT:EncryptionType. |
| | EncryptionType | Dieses optionale Element bestimmt das Verschlüsselungsverfahren.<br><br>Es MUSS das Verfahren XMLEnc: „http://www.w3.org/TR/xmlenc-core/" unterstützt werden, wenn das Dokument in CONSUMER:Base64XML übergeben wird und CMS: „urn:ietf:rfc:5652", wenn das Dokument in dss:Base64Data übergeben wird.<br><br>Die Verwendung dieses Elements ist aufgrund der impliziten Zuordnung der Verschlüsselungsverfahren zur Methode der Dokumentübergabe nicht erforderlich. |
| | CRYPT:UnprotectedProperties | Dieses optionale Element wird nur für das Verschlüsselungsverfahren CMS ausgewertet (zu verschlüsselndes Dokument ist in dss:Base64Data vorhanden).<br>Die Elemente ./UnprotectedProperties/Property/Value/CMSAttribute müssen base64/DER-kodiert ein vollständiges ASN.1-Attribute enthalten, definiert in [CMS# 9.1.AuthenticatedData Type]. Es muss bei der Erstellung des CMS-Containers unter "unauthAttrs" aufgenommen werden. Das zugehörige Element ./UnprotectedProperties/Property/Identifier wird nicht ausgewertet. |

| **Name** | **EncryptDocument** | |
|---|---|---|
| **Rückgabe** |  | |
| | `Status` | Enthält den Ausführungsstatus der Operation. |
| | `Document` | Enthält das verschlüsselte Dokument in Base64-codierter Form, wenn die Verschlüsselung erfolgreich durchgeführt wurde.<br>Im Fall XMLEnc wird das verschlüsselte XML-Dokument in `CONSUMER:Document/CONSUMER:Base64XML` zurückgegeben.<br>Im Fall CMS wird das verschlüsselte Dokument in `CONSUMER:Document/dss:Base64data` zurückgegeben. |
| **Vorbedingungen** | Keine | |
| **Nachbedingungen** | Keine | |

Vor der Verwendung für die Verschlüsselung MÜSSEN Zertifikate durch den Aufruf von PL_TUC_PKI_VERIFY_CERTIFICATE auf ihre Gültigkeit geprüft werden.
Abgelaufene oder gesperrte Zertifikate MÜSSEN von der Verwendung ausgeschlossen werden.

Das Verschlüsseln erfolgt durch Aufruf von PL_TUC_HYBRID_ENCIPHER {
Doc, das zu verschlüsselnde Dokument = `CONSUMER:Document;`
{Cert(i)}, „Menge der Empfänger-/Ziel-Zertifikate" = `RecipientKeys;`
Attribute, optionale, zusätzliche Attribute = `UnprotectedProperties;`
}
Wird ein Zertifikat per CertificateOnCard-Element referenziert, ist dieses vorher durch den HSMProxy zu extrahieren**[**<=, Basis-Consumer, Sich.techn. Eignung: Produktgutachten, funkt. Eignung: Test Produkt/FA**]**

Die Tabelle Tab_KeyReference_für_Encrypt/Decrypt wird entfernt:

| ~~**Karte**~~ | ~~Crypt (Wert)~~ | ~~**KeyReference (Encrypt)**~~ | ~~**KeyReference (Decrypt)**~~ |
|---|---|---|---|
| | | ~~In DF.ESIGN~~ | ~~In DF.ESIGN~~ |
| ~~SM-B (HSM)~~ | ~~RSA~~ | ~~EF.C.HCI.ENC.R2048~~ | ~~PrK.HCI.ENC.R2048~~ |
| | ~~ECC~~ | ~~EF.C.HCI.ENC.E256~~ | ~~PrK.HP.ENC.E256~~ |
| | ~~RSA_ECC~~ | ~~EF.C.HCI.ENC.R2048~~ ~~EF.C.HCI.ENC.E256~~ | ~~PrK.HCI.ENC.R2048~~ ~~PrK.HP.ENC.E256~~ |

Info: Lesende Kryptooperationen werden nicht geändert und bereits existierendes RSA-basiertes Material kann weiterhin vom Basis-Consumer verwendet werden. Daher:

- A_17515-03 wird nicht angepasst.
- A_24782 wird nicht angepasst.

In "8 Anhang B – Übersicht über die verwendeten Versionen" wird Tabelle 7 wie folgt geändert:

**Tabelle 7: Tab_Schema_Versionen Versionen der Schemas aus dem Namensraum des Basis- und KTR-Consumers**

| Schemas aus dem Namensraum des Basis- und KTR-Consumer „http://ws.gematik.de/consumer“ | | |
|---|---|---|
| **Name** | **Version** | **TargetNamespace** |
| CertificateService.wsdl | 3.0.0 | http://ws.gematik.de/consumer/CertificateService/WSDL/v3.0 |
| CertificateService.xsd | 3.0.0 | http://ws.gematik.de/consumer/CertificateService/v3.0 |
| CertificateServiceCommon.xsd | 2.0.0 | http://ws.gematik.de/consumer/CertificateServiceCommon/v2.0 |
| ConsumerCommon.xsd | 2.0.0 | http://ws.gematik.de/consumer/ConsumerCommon/v2.0 |
| EncryptionService.wsdl | 3.0.0 | http://ws.gematik.de/consumer/EncryptionService/WSDL/v3.0 |

| Schemas aus dem Namensraum des Basis- und KTR-Consumer „http://ws.gematik.de/consumer“ | | |
|---|---|---|
| EncryptionService.xsd | 3.0.0 | http://ws.gematik.de/consumer/EncryptionService/v3.0 |
| SignatureService.wsdl | 3.1.0 | http://ws.gematik.de/consumer/SignatureService/WSDL/v3.1 |
| SignatureService.xsd | 3.1.0 | http://ws.gematik.de/consumer/SignatureService/v3.1 |

# 4 Änderungen in gemSpec_Systemprozesse_dezTI

TIP1-A_6984-02 wird durch TIP1-A_6984-03 ersetzt:

**TIP1-A_6984-03 - Ablauf der hybriden Verschlüsselung eines Dokuments**
Produkttypen und Dienste der TI, die eine Plattformleistung PL_TUC_HYBRID_ENCYPHER umsetzen, MÜSSEN die Schritte zum Verschlüsseln eines gegebenen Dokuments in der angegebenen Reihenfolge durchführen:

| | **Teilschritt der hybriden Verschlüsselung** | **Teilergebnis** |
|---|---|---|
| 1 | Erzeugung eines symmetrischen Schlüssels gemäß BSI-TR-03116-1#3.5 Schlüsselerzeugung] und den Festlegungen in [gemSpec_Krypt#3.5.1 Hybride Verschlüsselung] ->$S_{symm}$ | Symmetrischer Schlüssel<br><br>Falls Symmetrischer Schlüssel nicht erzeugt werden kann<br>**=> Fehler** |
| 2 | Dokument Doc mit symmetrischem Schlüssel $S_{symm}$ verschlüsseln -> $Doc_{enc}$<br>Falls Doc ein XML-Dokument/Fragment ist: XMLEnc:<br>Es MÜSSEN die Vorgaben aus [gemSpec_Krypt#3.1.4] beachtet werden.<br>Sonst: CMS<br>Es MÜSSEN die Vorgaben aus [gemSpec_Krypt#3.5.1] beachtet werden. | symmetrisch verschlüsseltes Dokument |
| 3 | Für jedes Empfängerzertifikat Cert(i)<br>Schlüssel $S_{symm}$ mit öffentlichem Schlüssel $S_{public}$ der Empfängeridentität (liegt in Zertifikat Cert(i)) gemäß Vorgaben aus gemSpec_Krypt#3.1.5] (XML, RSA) bzw. [gemSpec_Krypt#3.5.2] (CMS, RSA) oder gemäß Vorgaben aus [gemSpec_Krypt#5.7] (XML/CMS, ECC) verschlüsseln -> $(S_{symm})_{enc(i)}$<br>Falls für einen Empfänger sowohl das RSA- als auch das ECC-Zertifikat vorliegt, wird der Schlüssel $S_{symm}$ ausschließlich mit öffentlichem Schlüssel $S_{public}$ der Empfängeridentität gemäß Vorgaben aus [gemSpec_Krypt#5.7] (XML/CMS, ECC) verschlüsselt. | pro Empfängerzertifikat: mit öffentlichem Schlüssel der Empfängeridentität verschlüsselter symmetrischer Schlüssel |

| | Teilschritt der hybriden Verschlüsselung | Teilergebnis |
|---|---|---|
| 4a | XMLEnc:<br>Alle verschlüsselten Dokumentenschlüssel $\{(S_{symm})_{enc(i)}\}$ als EncryptedKey und mit dem verschlüsselten Dokument $Doc_{enc}$ zu einem EncryptedData-Element gemäß [XML-Enc 1.1] zusammenfügen:<br>$Doc_{enc}$ + $\{(S_{symm})_{enc(i)}\}$ + Attribute -> D<br>Pro Empfängerzertifikat wird ein Element EncryptedKey/KeyInfo/X509Data/X509Certificate base64-kodiert und darin DER-kodiert abgelegt. | zusammengefügtes XML-ENC-EncryptedData-Element |
| 4b | CMS:<br>Alle verschlüsselten Dokumentenschlüssel $\{(S_{symm})_{enc(i)}\}$ und das verschlüsselte Dokument sind in einem Authenticated-Enveloped-Data Content Type gemäß [RFC-5083] und [RFC-5084] zusammenzuführen:<br>$Doc_{enc}$ + $\{(S_{symm})_{enc(i)}\}$ + Attribute -> D<br>Bei Verschlüsselung des „content-encryption key" wird „key transport" verwendet<br>Pro Empfängerzertifikat wird eine KeyTransRecipientInfo erzeugt, für RecipientIdentifier wird die Option IssuerAndSerialNumber verwendet<br>ContentType = OID {… authEnvelopedData}<br>= 1.2.840.113549.1.9.16.1.23 | zusammengefügtes CMS-Dokument |
| 5 | Rückmeldung an den Aufrufenden, entweder<br>1. OK + verschlüsseltes Dokument D oder<br>2. Fehler | |

**[**<=, Basis-Consumer, KTR-AdV, KTR-Consumer, funkt. Eignung: Herstellererklärung, Sich.techn. Eignung: Produktgutachten**]**

A_17377 wird durch A_17377-01 ersetzt:

**A_17377-01 - Aufrufparameter der nonQES Dokumenten-Signatur**
Produkttypen und Dienste der TI, die eine Plattformleistung PL_TUC_SIGN_DOCUMENT_nonQES umsetzen, MÜSSEN vom Nutzer den *IDENTIFIKATOR* des privaten Schlüssels der TI-Identität, das *SIGNATURVERFAHREN* gemäß ~~[gemSpec_Krypt#3.1.1 XML-Signaturen für nicht-qualifizierte Signaturen, 3.7 Signatur binärer Inhaltsdaten (Dokumente)] (RSA) bzw.~~ [gemSpec_Krypt#5.7.1 ECDSA-Signaturen im XML-Format, 5.7.2 ECDSA-Signaturen im CMS-Format] (ECC) sowie das zu signierende *DOKUMENT* und den *DOKUMENTENTYP* (XML, CMS) als Aufrufparameter entgegennehmen und in der Umsetzung verwenden. Das DOKUMENT MUSS gemäß DOKUMENTENTYP für die Signatur vorbereitet werden, dabei ist der zu signierende HASHWERT zu ermitteln.
Bei Nutzung einer SmartCard MUSS der *IDENTIFIKATOR* gemäß [gemSpecCardProxy#Konfigurationstabelle CardProxy] verwendet werden,

das *SIGNATURVERFAHREN sowie der HASHWERT* MÜSSEN als Aktionsparameter bzw. Eingangsparameter von *cardOperation* gemäß [gemSpec_CardProxy] verwendet werden. Bei Nutzung eines HSMs MUSS die Verwendung der genannten Aufrufparameter für Identity und Data gemäß [gemSpec_HSMProxy#logische Operation *sign* für Signaturen] erfolgen.
**[**<=, Basis-Consumer, FM_ePA_KTR, KTR-Consumer, funkt. Eignung: Herstellererklärung, Sich.techn. Eignung: Produktgutachten**]**

Info: Lesende Kryptooperationen werden nicht geändert und bereits existierendes RSA-basiertes Material kann weiterhin vom Basis-Consumer verwendet werden. Daher:

- TIP1-A_6987 muss nicht angepasst werden.
- A_17562 muss nicht angepasst werden.

# 5 Änderungen in GitHub - api-telematik

Die Schemadateien zum SignatureService werden gemäß GitHub PR https://github.com/gematik/api-telematik/pull/23 angepasst.