Gesellschaft für Telematikanwendungen der Gesundheitskarte mbH

Elektronische Gesundheitskarte und Telematikinfrastruktur

Übergreifende Spezifikation

Verwendung kryptographischer Algorithmen in der Telematikinfrastruktur

Version:	2.12.0
Revision:	72660
Stand:	18.12.2018
Status:	freigegeben
Klassifizierung:	öffentlich
Referenzierung:	gemSpec_Krypt

Dokumentinformationen

Änderungen zur Vorversion

Ergänzung von ePA-Inhalten. Diese sind gelb markiert.

Dokumentenhistorie

Version	Datum	Grund der Änderung, besondere Hinweise
1.4.0	03.07.08	freigegeben (für Release 2.3.4)
1.9.0	26.06.12	Kommentierung
1.10.0	13.09.12	Einarbeitung der Gesellschafterkommentare
2.0.0	15.10.12	bQS Kommentare eingearbeitet
2.1.0	06.06.13	Erweiterung im Rahmen der PP-Erstellung Konnektor (kryptographische Vorgaben für die SAK); Anpassung an das fortgeschriebene PP Konnektor ORS1 (BSI-CC-PP-0046), Konsistenz zur veränderten gemSpec_Kon herstellen
2.2.0	21.02.14	Losübergreifende Synchronisation
2.3.0	17.06.14	Entfernung des CBC-Modus bei der Dokumentenver- und -entschlüsselung gemäß P11-Änderungsliste
2.4.0	17.07.15	Einarbeitung Änderungen aus Errata 1.4.6
2.5.0	03.05.16	Anpassungen zum Online-Produktivbetrieb (Stufe 1)
2.6.0	24.08.16	Einarbeitung weiterer Kommentare
2.7.0	28.10.16	Anpassungen gemäß Änderungsliste
2.8.0	20.04.17	Start der Migration 120-Bit-Sicherheitsniveau kryptographische Verfahren in der TI (PKI der Kartengeneration 2.1), Anpassungen gemäß Änderungsliste
2.9.0	19.12.17	C_6260 (IPsec), C_6289 (qeS)
2.10.0	14.05.18	C_6195 (IPsec), C_6374 (ed. IPsec), C_6375 (TLS für SM), C_6399 (IPsec)
2.11.0	26.10.18	Veröffentlichung im Rahmen von Release 2.1.3
2.11.0	29.10.18	C_6639 (RSA 2048 Bit Zertifikate) etc., redaktionelle Aktualisierung der Anforderungen A_14653, A_15699 im Rahmen von Release 2.1.3
		ePA-Inhalte ergänzt
2.12.0	18.12.18	freigegeben

Inhaltsverzeichnis

1 Einführung

1.1 Zielsetzung und Einordnung des Dokuments

Die vorliegende übergreifende Spezifikation definiert Anforderungen an Produkte der TI bezüglich kryptographischer Verfahren. Diese Anforderungen sind als übergreifende Regelungen relevant für Interoperabilität und Verfahrenssicherheit.

Für die TI ist die Technische Richtlinie 03116 Teil 1 [BSI-TR-03116-1] normativ, d. h. nur dort aufgeführte kryptographische Verfahren dürfen von Produkten in der TI verwendet werden. Wenn mehrere unterschiedliche Produkttypen der TI zusammenarbeiten ist es bez. der Interoperabilität nicht sinnvoll wenn jeder beteiligter Produkttyp alle dort aufgeführten Verfahren umsetzen muss, da er vermuten muss die Gegenstelle beherrscht nur eine Teilmenge der dort aufgeführten Verfahren. Um einen gemeinsamen Nenner zu definieren, legt dieses Dokument für bestimmte Einsatzzwecke ein Mindestmaß an verpflichtend zu implementierenden Verfahren aus [BSI-TR-03116-1] fest, oftmals mit spezifischen Parametern. Ein Produkttyp ist frei, weitere Verfahren aus der [BSI-TR-03116-1] optional zu implementieren, kann sich jedoch nicht ohne Weiteres darauf verlassen, dass sein potentieller Kommunikationspartner diese auch beherrscht.

Dieses Dokument folgt den Konventionen der TR. Diese hat einen Betrachtungszeitraum von sechs bzw. sieben Jahren. Analog zu Kapitel 1 [BSI-TR-03116-1] bedeutet eine Aussage „Algorithmus X ist geeignet bis Ende 2023+“ generell nicht, dass Algorithmus X nach Ende 2023 nicht mehr geeignet ist, sondern lediglich dass über die Eignung nach Ende 2023 in der TR keine explizite Aussage gemacht wird und dass aus heutiger Sicht die weitere Eignung nicht ausgeschlossen ist. Aussagen über den Betrachtungszeitraum hinaus sind „mit einem höheren Maß an Spekulation verbunden“.

Bei neuen Erkenntnissen über die verwendeten kryptographischen Algorithmen, die zu einer Änderung der TR-03116-1 führen, wird eine Anpassung dieses Dokumentes erfolgen. Für Verwendungszwecke, bei denen bereits eine Migration zu stärkeren Algorithmen in Planung ist oder die Verwendung von Algorithmen unterschiedlicher Stärke zulässig ist, wird ein Ausblick gegeben, bis wann welche Algorithmen ausgetauscht sein müssen. Bei den Migrationsstrategien für kryptographische Algorithmen ist darauf zu achten, dass hinterlegte Objekte umzuschlüsseln sind bzw. die älteren Algorithmen (unter der Bedingung, dass sie sicherheitstechnisch noch geeignet sind) für eine gewisse Übergangsphase weiter unterstützt werden müssen und danach zuverlässig in den Komponenten deaktiviert werden müssen.

1.2 Zielgruppe

Das Dokument richtet sich an Hersteller und Anbieter von Produkten der TI, die kryptographische Objekte verwalten.

1.3 Geltungsbereich

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

Schutzrechts-/Patentrechtshinweis

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

1.4 Abgrenzung des Dokuments

Aufgabe des Dokumentes ist es nicht, eine Sicherheitsbewertung von kryptographischen Algorithmen vorzunehmen. Dieser Gesichtspunkt wird in [BSI-TR-03116-1] behandelt. Es werden lediglich die dort vorgegebenen Algorithmen weiter eingeschränkt, um die Herstellung der Interoperabilität zu unterstützen.

Es ist nicht Ziel dieses Dokumentes, den Prozess zum Austauschen von Algorithmen zu definieren, sondern lediglich den zeitlichen Rahmen für die Verwendbarkeit von Algorithmen festzulegen und somit auf den Bedarf für die Migration hinzuweisen.

1.5 Methodik

Anforderungen als Ausdruck normativer Festlegungen werden durch eine eindeutige ID sowie die dem RFC 2119 [RFC-2119] entsprechenden, in Großbuchstaben geschriebenen deutschen Schlüsselworte MUSS, DARF NICHT, SOLL, SOLL NICHT, KANN gekennzeichnet.

Sie werden im Dokument wie folgt dargestellt:

<AFO-ID> - <Titel der Afo>
Text / Beschreibung
[<=**]**

Dabei umfasst die Anforderung sämtliche zwischen Afo-ID und der Textmarke [<=] angeführten Inhalte.

2 Einsatzszenarioübergreifende Algorithmen

Nachfolgend werden grundlegende Festlegungen zur Verwendung von Algorithmen innerhalb der Telematikinfrastruktur getroffen. Diese Anforderungen sind unabhängig von den im nachfolgenden Kapitel definierten Einsatzszenarien und werden durch diese verwendet.

GS-A_3080 - asymmetrischen Schlüssel maximale Gültigkeitsdauer
Die Lebensdauer von asymmetrischen Schlüsseln und somit die in einem Zertifikat angegebene Gültigkeitsdauer SOLL maximal 5 Jahre betragen.
[<=]

2.1 Identitäten

Der Begriff „kryptographische Identität“ (nachfolgend nur noch als Identität bezeichnet) bezeichnet einen Verbund aus Identitätsdaten und einem kryptographischen Objekt, das bspw. im Rahmen einer Authentisierung und Authentifizierung verwendet werden kann. Im Allgemeinen handelt es sich um Schlüsselpaare, bestehend aus öffentlichem und privatem Schlüssel, sowie einem Zertifikat, das die Kombination aus Attributen und öffentlichem Schlüssel durch eine übergeordnete Instanz (CA – Certification Authority) bestätigt.

Bei den Algorithmenvorgaben für Identitäten muss u. a. spezifiziert werden:

- für welche Algorithmen und für welchen Verwendungszweck die Schlüssel verwendet werden (Bestimmte Verwendungszwecke schließen einander aus, bspw. dürfen nicht Signaturschlüssel für die Sicherung von Authentizität und Integrität von Dokumenten als Signaturschlüssel für beliebige Challenges im Rahmen einer Authentisierung verwendet werden.),
- welche Algorithmen für die Signatur des Zertifikates verwendet werden,
- mit welchen Algorithmen die OCSP-Responses signiert werden und
- wie die Zertifikate des OCSP-Responders signiert sind.

2.1.1 X.509-Identitäten

Eine X.509-Identität ist eine Identität gemäß Abschnitt 2.1, bei der ein X.509-Zertifikat [RFC-5280] verwendet wird.

Bei der Aufteilung von X.509-Identitäten wurden die Identitäten zunächst nach Gruppen für verschiedene Einsatzzwecke des Schlüssels unterteilt und diese bei Bedarf um einen notwendigen Einsatzkontext erweitert. Aus dieser Aufteilung ergibt sich die nachfolgend tabellarisch dargestellte Übersicht der Arten von X.509-Identitäten. Der exemplarische Einsatzort der Identitäten ist hierbei rein informativ, die Ausprägung wird in den Spezifikationen festgelegt, die eine kryptographische Identität benötigen.

Tabelle 1: Tab_KRYPT_001 Übersicht über Arten von X.509-Identitäten

Referenz	Gruppe	Kontext	Exemplarische Identitäten zur Verwendung (nicht vollständig)

2.1.1.1	Identitäten für die Erstellung von Signaturen	Identitäten für die Erstellung nicht-qualifizierter digitaler Signaturen	OSIG-Identität der SMC-B bzw. HSM-B
2.1.1.2		Identitäten für die Erstellung qualifizierter Signaturen	QES-Identität des HBA
2.1.1.5		Signaturidentitäten, die in den Diensten der TI-Plattform und den Fachdiensten zum Einsatz kommen.	Fachdienstsignatur Signatur durch zentrale Komponente der TI-Plattform Code-Signatur
2.1.1.3	Identitäten für die Client-Server-Authentifizierung	Identitäten für den Aufbau von TLS-Verbindungen	Fachdienst TLS – Server Fachdienst TLS – Client zentrale TI-Plattform TLS – Server zentrale TI-Plattform TLS – Client AUT-Identität der SMC-B AUT-Identität des Kartenterminals AUT-Identität des Anwendungskonnektors AUT-Identität der SAK AUT-Identität der eGK AUTN-Identität der eGK AUT-Identität des HBA
2.1.1.4		Identitäten für den Aufbau von IPsec-Verbindungen	ID.NK.VPN ID.VPNK.VPN
2.1.1.6	Verschlüsselungs-zertifikate	Identitäten, für die medizinische Daten verschlüsselt werden	ENC-Identität der eGK ENCV-Identität der eGK ENC-Identität des HBA ENC-Identität der SMC-B

Für den Aufbau der X.509-Zertifikate gelten die Vorgaben aus den jeweiligen Spezifikationen der X.509-Zertifikate.

2.1.1.1 Digitale nicht-qualifizierte elektronische Signaturen

GS-A_4357 - X.509-Identitäten für die Erstellung und Prüfung digitaler nicht-qualifizierter elektronischer Signaturen

Alle Produkttypen, die X.509-Identitäten bei der Erstellung oder Prüfung digitaler nicht-qualifizierter elektronischer Signaturen verwenden, MÜSSEN die in Tab_KRYPT_002 aufgeführten Algorithmen unterstützen und die Tabellenvorgaben erfüllen.
Produkttypen, die Zertifikate (X.509-Identitäten) auf Basis der Schlüsselgeneration „ECDSA“ ausstellen (vgl. Abschnitt 5.1) oder verwenden, MÜSSEN die in Tab_KRYPT_002a aufgeführten Algorithmen und die Tabellenvorgaben erfüllen.
[<=**]**

Tabelle 2: Tab_KRYPT_002 Algorithmen für X.509-Identitäten zur Erstellung nicht-qualifizierter Signaturen für die Schlüsselgeneration „RSA“

Anwendungsfall	Vorgaben
Art und Kodierung des öffentlichen Schlüssels	RSA (OID 1.2.840.113549.1.1.1) zu verwendende Schlüssellänge: 2048 Bit, zulässig bis Ende 2023 [BSI-TR-03116-1], vgl. auch A_15590
Signatur eines Zertifikats Signatur einer OCSP-Response Signatur eines OCSP-Responder-Zertifikates Signatur einer CRL Signatur des Zertifikats das Basis der Signaturprüfung einer CRL ist	sha256withRSAEncryption (OID 1.2.840.113549.1.1.11) zu verwendende Schlüssellänge: 2048 Bit, zulässig bis Ende 2023 [BSI-TR-03116-1], vgl. auch A_15590

A_15590 - Zertifikatslaufzeit bei Erstellung von X.509-Zertifikaten mit RSA 2048 Bit
Ein TSP-X.509-nonQES, der X.509-Zertifikate erstellt auf Basis der Schlüsselgeneration „RSA“ (d. h., für den die Vorgaben aus Tab_KRYPT_002 gelten), MUSS das Ende der Zertifikatsgültigkeitsdauer für das auszustellende Zertifikat unabhängig von der in Tab_KRYPT_002 festgelegten Endedaten der Zulässigkeit der verwendeten RSA-Schlüssellängen festlegen.**[**<=**]**

Erläuterung: Die technische Durchsetzung des Endes der Zulässigkeit von RSA mit weniger als 3000 Bit Schlüssellänge in X.509-Zertifikaten erfolgt durch die Herausnahme der entsprechenden RSA-basierten Sub-CA-Zertifikate aus der TSL zum Zeitpunkt des Ablaufens der Zulässigkeit (gemäß TIP1-A_2062). Ein TSP muss bez. der Zertifikatsgültigkeitsdauer der von ihm ausgegebenen Zertifikate das nach Spezifikationslage definierte Verhalten zeigen (i. A. Zertifikatsgültigkeitsdauer der ausgegebenen Zertifikate von 5 Jahren). Ein TSP kann auch mit dem Kartenherausgeber beliebige Gültigkeitsdauern unter 5 Jahren für die Laufzeit der vom TSP ausgegebenen Zertifikate vereinbaren.

Tabelle 3: Tab_KRYPT_002a Algorithmen für X.509-Identitäten zur Erstellung nicht-qualifizierter Signaturen für die Schlüsselgeneration „ECDSA“

Anwendungsfall	Vorgabe
Art und Kodierung des öffentlichen Schlüssels	ecPublicKey {OID 1.2.840.10045.2.1} auf der Kurve brainpoolP256r1 [RFC-5639#3.4, brainpoolP256r1] zulässig bis Ende 2023+ Die Kodierung des öffentlichen Punkt erfolgt nach [RFC5480, Abschnitt 2], vgl. Beispiel in Abschnitt 5.2) Der privater Schlüssel muss zufällig und gleichverteilt aus {1, …, q} gewählt werden. (q ist die Ordnung des Basispunkts und ceil($\log_2$ q)=256).

Signatur eines Zertifikats Signatur einer OCSP-Response Signatur eines OCSP-Responder-Zertifikates Signatur einer CRL Signatur des Zertifikats das Basis der Signaturprüfung einer CRL ist	ecdsa-with-SHA256 [RFC-3279] {OID 1.2.840.10045.4.3.2} auf der Kurve brainpoolP256r1 [RFC-5639#3.4, brainpoolP256r1] zulässig bis Ende 2023+ vgl. Beispiel in Abschnitt 5.2 Der privater Schlüssel muss zufällig und gleichverteilt aus {1, …, q} gewählt werden. (q ist die Ordnung des Basispunkts und ceil($\log_2$ q)=256).

Aktuell werden in der TI CRLs ausschließlich im Rahmen des IPsec-Verbindungsaufbaus (Verbindung der Konnektoren in die TI) verwendet.

Für die maximale Gültigkeitsdauer der Zertifikate gilt die Anforderung [GS-A_3080].

2.1.1.2 Qualifizierte elektronische Signaturen

GS-A_4358 - X.509-Identitäten für die Erstellung und Prüfung qualifizierter elektronischer Signaturen

Alle Produkttypen, die X.509-Identitäten für die Erstellung oder Prüfung von qualifizierten elektronischen Signaturen verwenden, MÜSSEN mindestens alle in Tabelle Tab_KRYPT_003 aufgeführten Algorithmen unterstützen und die Tabellenvorgaben erfüllen.

TSP-X.509-QES, die qualifizierte Zertifikate (X.509-Identitäten) auf Basis der Schlüsselgeneration „ECDSA“ (vgl. Abschnitt 5.1) erstellen oder verwenden MÜSSEN die in Tab_KRYPT_003a aufgeführten Algorithmen und die Tabellenvorgaben erfüllen.

[<=]

Tabelle 4: Tab_KRYPT_003 Algorithmen für X.509-Identitäten zur Erstellung qualifizierter elektronischer Signaturen für die Schlüsselgeneration „RSA“

Anwendungsfälle	Vorgaben
Signatur des VDA-Zertifikats	Nachdem die eIDAS-Verordnung das Signaturgesetz vollständig abgelöst hat, steht es einem VDA frei zu entscheiden welche Signatur (bspw. signiert von einer beliebigen VDA-internen CA) sein VDA-Zertifikat haben soll. Insbesondere kann die Signatur mit einem Nicht-RSA-Verfahren erstellt werden. Eine auswertende Komponente muss mit beliebigen (also auch nicht-RSA basierten) Signaturen eines VDA-Zertifikats umgehen können (bspw. Signatur des VDA-Zertifikats nicht auswerten, Authentizität und Integrität des Zertifikats wird über die Vertrauensliste sichergestellt).
Art und Kodierung des öffentlichen EE-Schlüssels	RSA-Signaturvariante: **Entweder** OID 1.2.840.113549.1.1.1 (rsaEncryption) (zulässig bis Ende 2022 [SOG-IS-2018]) **oder** OID 1.2.840.113549.1.1.10 (id-RSASSA-PSS) [RFC-5756]. (ohne zeitliche Beschränkung der Zulässigkeit [SOG-IS-2018]) Die Auswahl obliegt dem EE-Zertifikatsausgebenden VDA.

	RSA-Schlüssellänge: zu verwendende Schlüssellänge: 2048 Bit, zulässig bis Ende 2024 [SOG-IS-2018]
Signatur eines Zertifikats, Signatur einer OCSP-Response oder Signatur eines OCSP-Responder-Zertifikates	**Entweder** sha256withRSAEncryption (OID 1.2.840.113549.1.1.11) (zulässig bis Ende 2022 [SOG-IS-2018]) **oder** id-RSASSA-PSS (1.2.840.113549.1.1.10) [RFC-5756] (ohne zeitliche Beschränkung der Zulässigkeit [SOG-IS-2018]) zu verwendende Schlüssellänge: 2048 Bit, zulässig bis Ende 2024 [SOG-IS-2018] Die Hashfunktion für die Hashwertberechnung der TBSCertificate-Datenstruktur MUSS eine nach [SOG-IS-2018] zulässige Hashfunktion („Agreed Hash Function“) sein. Als Hashfunktion SOLL SHA-256 [FIPS-180-4] verwendet werden. Als MGF MUSS MGF1 [PKCS#1] verwendet werden. Die innerhalb der MGF1 verwendete Hashfunktion MUSS die gleiche Hashfunktion sein, wie die Hashfunktion der Hashwertberechnung der TBSCertificate-Datenstruktur. (Dies entspricht der Empfehlung aus [RFC-5756] bzw. [RFC-4055, 3.1] und dient der Komplexitätsreduktion.) Die Saltlänge MUSS mindestens 256 Bit betragen.(Die Maximallänge des Salts ergibt sich nach [PKCS#1] in Abhängigkeit von der Länge des Moduls.)

Tabelle 5: Tab_KRYPT_003a Algorithmen für X.509-Identitäten zur Erstellung qualifizierter Signaturen für die Schlüsselgeneration „ECDSA“

Anwendungsfall	Vorgabe
Signatur des VDA-Zertifikats	Nachdem die eIDAS-Verordnung das Signaturgesetz vollständig abgelöst hat, steht es einem VDA frei zu entscheiden welche Signatur (bspw. signiert von einer beliebigen VDA-internen CA) sein VDA-Zertifikat haben soll. Insbesondere kann die Signatur mit einem Nicht-ECDSA-Verfahren erstellt werden. Eine auswertende Komponente muss mit beliebigen (also auch nicht-ECDSA basierten) Signaturen eines VDA-Zertifikats umgehen können (bspw. Signatur des VDA-Zertifikats nicht auswerten, Authentizität und Integrität des Zertifikats wird über die Vertrauensliste sichergestellt).
Art und Kodierung des öffentlichen EE-Schlüssels	ecPublicKey {OID 1.2.840.10045.2.1} auf der Kurve brainpoolP256r1 [RFC-5639#3.4, brainpoolP256r1] zulässig bis Ende 2023+ Die Kodierung des öffentlichen Punkt erfolgt nach [RFC5480, Abschnitt 2], vgl. Beispiel in Abschnitt 5.2). Der privater Schlüssel muss zufällig undgleichverteilt aus {1, …, q} gewählt werden. (q ist die Ordnung des Basispunkts und ceil($\log_2$ q)=256).

Signatur eines Zertifikats, Signatur einer OCSP-Response oder Signatur eines OCSP-Responder-Zertifikates	ecdsa-with-SHA256 [RFC-3279] {OID 1.2.840.10045.4.3.2} auf Kurve der brainpoolP256r1 [RFC-5639#3.4, brainpoolP256r1] zulässig bis Ende 2023+ vgl. Beispiel in Abschnitt 5.2

2.1.1.3 TLS-Authentifizierung

GS-A_4359 - X.509-Identitäten für die Durchführung einer TLS-Authentifizierung
Alle Produkttypen, die X.509-Identitäten für eine TLS-Authentifizierung verwenden, MÜSSEN alle in Tab_KRYPT_002 aufgeführten Algorithmen unterstützen und die Tabellenanforderungen erfüllen.
Produkttypen die Zertifikate (X.509-Identitäten) auf Basis der Schlüsselgeneration „ECDSA“ ausstellen (vgl. Abschnitt 5.1) oder verwenden, MÜSSEN die in Tab_KRYPT_002a aufgeführten Algorithmen und die Tabellenvorgaben erfüllen.
[<=**]**

2.1.1.4 IPsec-Authentifizierung

GS-A_4360 - X.509-Identitäten für die Durchführung der IPsec-Authentifizierung
Alle Produkttypen, die X.509-Identitäten für eine IPsec-Authentifizierung verwenden, MÜSSEN alle in Tab_KRYPT_002 aufgeführten Algorithmen unterstützen und die Tabellenanforderungen erfüllen.
Produkttypen die Zertifikate (X.509-Identitäten) auf Basis der Schlüsselgeneration „ECDSA“ ausstellen (vgl. Abschnitt 5.1) oder verwenden, MÜSSEN die in Tab_KRYPT_002a aufgeführten Algorithmen und die Tabellenvorgaben erfüllen.
[<=**]**

2.1.1.5 Digitale Signaturen durch TI-Komponenten

GS-A_4361 - X.509-Identitäten für die Erstellung und Prüfung digitaler Signaturen
Alle Produkttypen, die X.509-Identitäten verwenden, die zur Erstellung und Prüfung digitaler Signaturen in Bezug auf TI-Komponenten (technische X.509-Zertifikate) genutzt werden, MÜSSEN alle in Tab_KRYPT_002 aufgeführten Algorithmen unterstützen und die Tabellenanforderungen erfüllen.
Produkttypen die Zertifikate (X.509-Identitäten) auf Basis der Schlüsselgeneration „ECDSA“ ausstellen (vgl. Abschnitt 5.1) oder verwenden, MÜSSEN die in Tab_KRYPT_002a aufgeführten Algorithmen und die Tabellenvorgaben erfüllen.
[<=**]**

2.1.1.6 Verschlüsselung

GS-A_4362 - X.509-Identitäten für Verschlüsselungszertifikate
Alle Produkttypen, die X.509-Identitäten für die Verschlüsselung (Verschlüsselungszertifikate) verwenden, MÜSSEN alle in Tab_KRYPT_002 aufgeführten Algorithmen unterstützten und die Tabellenanforderungen erfüllen.
Produkttypen die Zertifikate (X.509-Identitäten) auf Basis der Schlüsselgeneration „ECDSA“ ausstellen (vgl. Abschnitt 5.1) oder verwenden, MÜSSEN die in Tab_KRYPT_002a aufgeführten Algorithmen und die Tabellenvorgaben erfüllen.
[<=**]**

2.1.2 CV-Identitäten

CV-Identitäten werden für die Authentifizierung zwischen Karten verwendet.

2.1.2.1 CV-Zertifikate G1

GS-A_4363 - CV-Zertifikate G1
Alle Produkttypen, die CV-Zertifikate der Kartengeneration G1 erstellen oder prüfen, MÜSSEN die in Tab_KRYPT_004 aufgeführten Algorithmen verwenden und die Tabellenanforderungen erfüllen.
[<=]

Tabelle 6: Tab_KRYPT_004 Algorithmen für CV-Zertifikate

Algorithmen Typ	Algorithmus	Schlüssellänge
über das Zertifikat bestätigtes Schlüsselpaar	authS_ISO9796-2 Withrsa_sha256_mutual (OID 1.3.36.3.5.2.4)	2048 Bit bis Ende 2018
Signatur des Endnutzerzertifikats	sigS_ISO9796-2Withrsa_sha256 (OID 1.3.36.3.4.2.2.4)	2048 Bit bis Ende 2018

Für die maximale Gültigkeitsdauer der Zertifikate gilt die Anforderung [GS-A_3080].

Das verwendete Signaturverfahren ISO-9796-2 DS1 ist nach [BSI-TR-03116-1] in der TI nur noch bis Ende 2018 zulässig. Damit ist Ende 2018 eine obere Schranke für das Ende der G1-Karten.

2.1.2.2 CV-Certification-Authority (CV-CA) Zertifikat G1

GS-A_4364 - CV-CA-Zertifikate G1
Alle Produkttypen, die CV-CA-Zertifikate der Kartengeneration G1 erstellen oder prüfen, MÜSSEN die in Tab_KRYPT_005 aufgeführten Algorithmen verwenden und die Tabellenanforderungen erfüllen.
[<=]

Tabelle 7: Tab_KRYPT_005 Algorithmen für CV-CA-Zertifikate

Algorithmen Typ	Algorithmus	Schlüssellänge
über das Zertifikat bestätigtes Schlüsselpaar	sigS_ISO9796-2Withrsa_sha256 (OID 1.3.36.3.4.2.2.4)	2048 Bit bis Ende 2018
Signatur des CA-Zertifikates	sigS_ISO9796-2Withrsa_sha256 (OID 1.3.36.3.4.2.2.4)	2048 Bit bis Ende 2018

Für die maximale Gültigkeitsdauer der Zertifikate gilt die Anforderung [GS-A_3080].

Das verwendete Signaturverfahren ISO-9796-2 DS1 ist nach [BSI-TR-03116-1] in der TI nur noch bis Ende 2018 zulässig. Damit ist Ende 2018 eine obere Schranke für das Ende der G1-Karten.

2.1.2.3 CV-Zertifikate G2

GS-A_4365 - CV-Zertifikate G2
Alle Produkttypen, die CV-Zertifikate der Kartengeneration G2 erstellen oder prüfen, MÜSSEN die in Tab_KRYPT_006 aufgeführten Algorithmen verwenden und die Tabellenanforderungen erfüllen.
[<=]

Tabelle 8: Tab_KRYPT_006 Algorithmen für CV-Zertifikate

Algorithmen Typ	Algorithmus	Schlüssellänge
über das Zertifikat bestätigtes Schlüsselpaar	**Authentisierung ohne Sessionkey-Aushandlung** [RFC-5639#3.4, brainpoolP256r1] ecdsa-with-SHA256 {OID 1.2.840.10045.4.3.2} **Authentisierung mit Sessionkey-Aushandlung** [RFC-5639#3.4, brainpoolP256r1] authS_gemSpec-COS-G2_ecc-with-sha256 {OID 1.3.36.3.5.3.1}	256 Bit bis Ende 2023+
Signatur des Endnutzerzertifikats	[RFC-5639#3.4, brainpoolP256r1] ecdsa-with-SHA256 {OID 1.2.840.10045.4.3.2}	256 Bit bis Ende 2023+

Für die maximale Gültigkeitsdauer der Zertifikate gilt die Anforderung [GS-A_3080].

2.1.2.4 CV-Certification-Authority (CV-CA) Zertifikat G2

GS-A_4366 - CV-CA-Zertifikate G2
Alle Produkttypen, die CV-CA-Zertifikate der Kartengeneration G2 erstellen oder prüfen, MÜSSEN dien Tab_KRYPT_007 aufgeführten Algorithmen verwenden und die Tabellenanforderungen erfüllen.
[<=]

Tabelle 9: Tab_KRYPT_007 Algorithmen für CV-CA-Zertifikate

Algorithmen Typ	Algorithmus	Schlüssellänge
über das Zertifikat bestätigtes Schlüsselpaar	[RFC-5639#3.4, brainpoolP256r1] ecdsa-with-SHA256 {OID 1.2.840.10045.4.3.2}	256 Bit bis Ende 2023+
Signatur des CA-Zertifikates	[RFC-5639#3.4, brainpoolP256r1] ecdsa-with-SHA256 {OID 1.2.840.10045.4.3.2}	256 Bit bis Ende 2023+

Für die maximale Gültigkeitsdauer der Zertifikate gilt die Anforderung [GS-A_3080].

2.2 Zufallszahlengeneratoren

GS-A_4367 - Zufallszahlengenerator
Alle Produkttypen, die Zufallszahlen generieren, MÜSSEN die Anforderungen aus [BSI-TR-03116-1#3.48 Erzeugung von Zufallszahlen] erfüllen.
[<=]

2.3 Hilfestellung bei der Umsetzung (Zufallsgeneratoren)

(Hinweis: dies ist das ehemalige „Kapitel 5.2.4 Hilfestellung bei der Umsetzung der Anforderungen“. Der Text in diesem Abschnitt entstand in enger Abstimmung mit dem BSI auf Gesellschafterwunsch.)

Die Sicherheit eines deterministischen Zufallszahlengenerators (DRNGs) hängt maßgeblich von drei Faktoren ab:

- von der Entropie des Seeds,
- vom algorithmischen Anteil (generelles Design) und
- dem Schutz des inneren Zustands (und der zur Ausgabe vorgesehenen Zufallszahlen).

Der Nachweis, dass der algorithmische Anteil eines DRNGs den Anforderungen einer bestimmten Funktionalitätsklasse genügt, kann schwierig und aufwändig sein. Deshalb wurde das BSI gebeten, die DRNGs in [FIPS-186-2+CN1] und [ANSI-X9.31] in Bezug auf die kryptographische Güte ihres algorithmischen Anteils zu bewerten.

Das Ergebnis ist:

A) [FIPS-186-2+CN1]: Lässt man in dem DRNG aus Appendix 3.1 (S. 16f.) in Schritt 3c bzw. in dem DRNG aus Algorithmus 1 (Change Notice 1, S. 72f.) in Schritt 3.3 den Term "mod q" weg, so werden gleich verteilt 160-Bit Zufallszahlen bzw. 320-Bit Zufallszahlen erzeugt (vgl. Abschnitt „General Purpose Random Number Generation“ (Change Notice 1, S. 74)).
Beide DRNGs sind dann

1. algorithmisch geeignet für die Klasse K4 [AIS-20-1999] und
2. erfüllen die algorithmischen Anforderungen aus DRG.3 [AIS-20].

Ob eine konkrete Implementierung eines dieser DRNG bspw. Teil der Klasse DRG.3 ist, bleibt im Einzelfall zu prüfen, da dazu u. a. auch Fragen über die Initialisierung zu beantworten sind (vgl. (DRG.3.1) [KS-2011]).

Das BSI empfiehlt bei den Zufallsgeneratoren aus [FIPS-186-2+CN1] nach Möglichkeit SHA-256 [FIPS-180-4] anstatt SHA-1 zu verwenden. Folgt man der Empfehlung, so ist der Algorithmus dementsprechend zu adaptieren.

B) [ANSI-X9.31]: Der Zufallsgenerator aus Appendix A.2.4 ist

(1) algorithmisch geeignet für die Klasse K3 [AIS-20-1999] und

(2) erfüllt die algorithmischen Anforderungen aus DRG.2 [AIS-20].

2.4 Schlüsselerzeugung und Schlüsselbestätigung

GS-A_4368 - Schlüsselerzeugung
Alle Produkttypen, die Schlüssel erzeugen, MÜSSEN die Anforderungen aus [BSI-TR-03116-1#3.~~5~~9 Schlüsselerzeugung] erfüllen.**[<=]**

Hinweis: im Rahmen der Sicherheitszertifizierung von Komponenten, wie bspw. des Konnektors, wird dies überprüft.

GS-A_5021 - Schlüsselerzeugung bei einer Schlüsselspeicherpersonalisierung
Ein Herausgeber von Sicherheitsmodulen für kryptographisches Schlüsselmaterial, welche in der TI genutzt werden (also bspw. eGK, SMC-B, HSM-B, SMC-KT und HBA), MUSS sicherstellen, dass auf dem Sicherheitsmodul gespeicherten Schlüssel die Anforderungen aus [BSI-TR-03116-1#3.5 Schlüsselerzeugung] erfüllen.
[<=]

Hinweis: Dies ist eine Anforderung an Kartenherausgeber, die so sicherstellen müssen, dass das in den Sicherheitsmodulen (also auch HSM-B) zur Verfügung stehende kryptographische Schlüsselmaterial geeignet ist Daten mit sehr hohem Schutzbedarf schützen zu können. (siehe auch Kapitel 4.4)

GS-A_5338 - HBA/SMC-B – Erzeugung asymmetrischer Schlüsselpaare auf der jeweiligen Karte selbst
Ein Kartenherausgeber oder, falls der Kartenherausgeber einen Dritten mit der Kartenpersonalisierung beauftragt, der Kartenpersonalisierer für HBA oder SMC-B MUSS sicherstellen, dass bei der Personalisierung der Karten HBA und SMC-B alle asymmetrischen Schlüsselpaare, bei denen die privaten Schlüssel auf der Karte gespeichert werden, auf der Karte erzeugt werden.
[<=]

Aufgrund des geringeren Mengengerüsts bei HBA und SMC-B ist dort die On-Card-Generierung der entsprechenden Schlüsselpaare möglich. Somit (vgl. auch [PP-0082, FPT_EMS.1] ist technisch sichergestellt, dass keine Kopie der privaten Schlüssel außerhalb der Chipkarte existiert (Kontext: Ende-zu-Ende-Verschlüsselung von medizinischen Daten).

GS-A_5386 - kartenindividuelle geheime und private Schlüssel G2-Karten
Ein Kartenherausgeber, der G2-Karten herausgibt, MUSS sicherstellen, dass bei der Personalisierung der Karten alle für eine Karte zu personalisierenden privaten und geheimen Schlüssel kartenindividuell sind. Bei Beauftragung eines Dritten mit der Schlüsselerzeugung ist dies durch den Dritten sicherzustellen.
Falls symmetrische Schlüssel (bspw. SK.CMS.AES128) nicht pro Karte zufällig erzeugt werden, sondern mit einem Schlüsselableitungsverfahren erzeugt werden, so MUSS der Kartenherausgeber sicherstellen, dass

1. das verwendete Schlüsselableitungsverfahren (KDF) unumkehrbar und nicht-vorhersagbar ist (Hilfestellung: Beispiele in [gemSpec_Krypt, 2.4 und 3.4]).
2. der Masterkey (Key Derivation Key (KDK)) GS-A_4368 erfüllt (insbesondere Entropie-Vorgaben). Der KDK MUSS eine Mindestentropie von 120 Bit besitzen.

[<=]

Für private Schlüssel bei HBA und SMC-B wird die kartenindividuelle Erzeugung und Personalisierung durch GS-A_5338 technisch sichergestellt. Je nach verwendetem COS, insbesondere dessen spezifischen Personalisierungsverfahrens, kann es sein, dass ein Kartenherausgeber symmetrische Schlüssel aus technischen Gründen personalisieren muss, obwohl er später nicht plant mit diesen Schlüsseln bspw. im Rahmen eines CMS

zu arbeiten. Es ist sicherheitskritisch, dass auch diese symmetrischen Schlüssel ebenfalls die Anforderungen GS-A_5021 bzw. GS-A_4368 erfüllen.

Als geeignete Schlüsselableitungsverfahren (KDF) für die Erzeugung von kartenindividuellen Schlüssel sind bspw. folgende Verfahren geeignet:

- alle Verfahren aus [NIST-SP-800-108] mittels CMAC [NIST-SP-800-38B],
- alle Verfahren aus [NIST-SP-800-56-A] bzw. [NIST-SP-800-56-B] mittels jeder nach [BSI-TR-03116-1] zulässigen Hashfunktion,
- alle Verfahren aus [NIST-SP-800-56C] mittels CMAC [NIST-SP-800-38B] oder eines HMAC, der auf einer nach [BSI-TR-03116-1] zulässigen Hashfunktion basiert,
- das Verfahren nach [ANSI-X9.63, Abschnitt 5.6.3] mittels jeder nach [BSI-TR-03116-1] zulässigen Hashfunktion.

~~2.5 Padding~~

~~2.5.1 Zufalls-Padding für Blockchiffren bei XML-Verschlüsselung~~

~~*Nach dem Umstieg vom Betriebsmodus CBC auf den GCM (symmetrische Verschlüsselung) sind die ehemaligen Vorgaben in diesem Abschnitt obsolet.*~~

3 Konkretisierung der Algorithmen für spezifische Einsatzszenarien

In den nachfolgenden Abschnitten werden die kryptographischen Algorithmen für verschiedene Einsatzszenarien spezifiziert. In diesem Zusammenhang sind ausschließlich die kryptographischen Aspekte der Einsatzszenarien relevant.

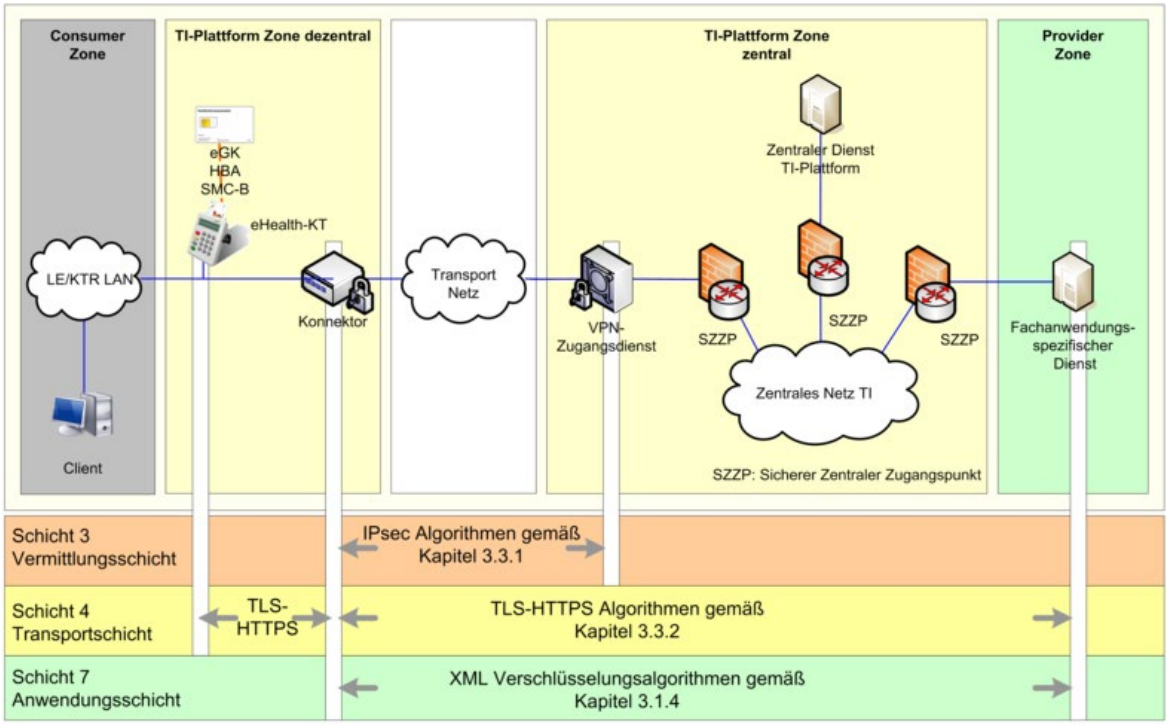

Abbildung 1: Verwendung von Algorithmen nach Zonen und OSI-Schicht

Abbildung 1 stellt beispielhaft die für die Vertraulichkeit von medizinischen Daten relevanten Algorithmen auf den verschiedenen OSI-Schichten in einer Übersicht dar. Es besteht in dieser Abbildung kein Anspruch auf Vollständigkeit.

3.1 Kryptographische Algorithmen für XML-Dokumente

GS-A_4370 - Kryptographische Algorithmen für XML-Dokumente
Alle Produkttypen, die XML-Dokumente

- verschlüsseln, MÜSSEN dies mittels CMS (PKCS#7) oder XMLEnc durchführen,
- signieren, MÜSSEN dies mittels CMS (PKCS#7) oder XMLDSig durchführen.

[<=]

XML-Signaturen sind bezüglich der verwendeten Algorithmen selbst beschreibend, die für die Erstellung einer Signatur verwendeten Algorithmen sind in der Signatur aufgeführt.

Zur vollständigen Spezifikation der Algorithmen für XML-Signaturen müssen für alle Signaturbestandteile Algorithmen spezifiziert werden. Die nachfolgenden Abschnitte wählen aus der Menge der zulässigen Algorithmen die jeweiligen Algorithmen für die einzelnen Einsatzszenarien aus.

Die Referenzierung von Algorithmen in XML-Signaturen und XML-Verschlüsselungen erfolgt nicht wie in Zertifikaten oder Signaturen binärer Daten über OIDs sondern über URIs. Die URIs der Algorithmen dienen als eindeutige Identifier und nicht dazu, dass unter der jeweils angegebenen URI die Beschreibung zu finden ist.

Tabelle 10: Tab_KRYPT_008 Beispiele für solche Algorithmen-URIs

Algorithmen Identifier	Erläutert in
http://www.w3.org/2001/04/xmlenc#aes256-cbc	[XMLEnc]
http://www.w3.org/2001/04/xmlenc#rsa-1_5	[XMLEnc]
http://www.w3.org/2001/04/xmlenc#sha256	[XMLDSig]
http://www.w3.org/2000/09/xmldsig#enveloped-signature	[XMLDSig]
http://www.w3.org/2001/04/xmldsig-more#rsa-sha256	[RFC-4051] bzw. [RFC-6931]
http://www.w3.org/2001/10/xml-exc-c14n#	[XMLCan_V1.0]
http://www.w3.org/2009/xmlenc11#aes256-gcm	[XMLEnc-1.1]
http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1	[RFC-6931]

3.1.1 XML-Signaturen für nicht-qualifizierte Signaturen

GS-A_4371 - XML-Signaturen für nicht-qualifizierte Signaturen
Alle Produkttypen, die XML-Signaturen für nicht-qualifizierte Signaturen erzeugen oder prüfen, MÜSSEN die Algorithmen und Vorgaben der Tabelle Tab_KRYPT_009 erfüllen.
[<=]

Tabelle 11: Tab_KRYPT_009 Algorithmen für die Erzeugung von nicht-qualifizierten elektronischen XML-Signaturen

Signaturbestandteil	Beschreibung	Algorithmus	Anmerkung
Signaturstandard	Signaturstandard	ETSI TS 101 903 V1.4.2 (2010-12) Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES) [ETSI-XAdES]	Die Verwendung des Standards ist für die Signatur von XML-Dokumenten

			verpflichtend, die nicht über CMS (PKCS#7) signiert werden.
kryptographisches Signaturverfahren	Algorithmus für die Berechnung des Nachrichten Digest und die Verschlüsselung mit dem privaten Schlüssel	RSASSA-PSS mit SHA256 bis nach Ende 2021+ verwendbar (Ende des Betrachtungshorizonts) (Hinweis: siehe Abschnitt 4.1)	Die Verwendung des Algorithmus ist verpflichtend. Alle hier aufgeführten Signaturverfahren müssen von einer Signaturprüfenden Komponente überprüfbar sein.
DigestMethod	Methode zur Berechnung eines Digest der zu signierenden Bereiche	SHA-256 Die [XMLDSig] konforme Bezeichnung lautet: http://www.w3.org/2001/04/xmlenc#sha256	Die Verwendung des Algorithmus ist verpflichtend.
Kryptographisches Token	Kryptographisches Token für die Signatur, bestehend aus einem privaten Schlüssel und einem zugehörigen X.509-Zertifikat	Identitäten gemäß einem der folgenden Abschnitte 2.1.1.1	Die Auswahl des kryptographischen Tokens ist von dem jeweiligen Einsatzzweck abhängig.

3.1.2 XML-Signaturen für qualifizierte elektronische Signaturen

GS-A_4372 - XML-Signaturen für qualifizierte elektronische Signaturen
Alle Produkttypen, die XML-Signaturen für qualifizierte elektronische Signaturen erzeugen oder prüfen, MÜSSEN die Vorgaben der Tabelle Tab_KRYPT_010 erfüllen.
[<=]

Tabelle 12: Tab_KRYPT_010 Algorithmen für qualifizierte XML-Signaturen

Signaturbestandteil	Beschreibung	Algorithmus	Anmerkung
Signaturstandard	Signaturstandard	ETSI TS 101 903 V1.4.2 (2010-12) Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES) [ETSI-XAdES]	Die Verwendung des Standards ist für die Signatur von XML-Dokumenten verpflichtend, die

			nicht über CMS (PKCS#7) signiert werden.
kryptographisches Signaturverfahren	Algorithmus für die Berechnung des Nachrichten-Digest und die Verschlüsselung mit dem privaten Schlüssel	RSASSA-PSS mit SHA256 bis nach Ende 2023+ verwendbar (Ende des Betrachtungshorizonts) (Hinweis: siehe Abschnitt 4.1)	Der Algorithmus muss für alle qualifizierten Signaturen verwendet werden. Alle hier aufgeführten Signaturverfahren müssen von einer Signaturprüfenden Komponente überprüfbar sein.
DigestMethod	Methode zur Berechnung eines Digest der zu signierenden Bereiche	SHA-256 Die [XMLDSig] konforme Bezeichnung lautet: http://www.w3.org/2001/04/xmlenc#sha256	Der Algorithmus muss für alle qualifizierten Signaturen verwendet werden.
Kryptographisches Token	Kryptographisches Token für die Signatur, bestehend aus einem privaten Schlüssel und einem zugehörigen X.509-Zertifikat	Identitäten gemäß dem folgenden Abschnitt 2.1.1.2	Es darf nur eine Identität, die den Ansprüchen qualifizierter Signaturen entspricht, verwendet werden.

3.1.3 Webservice Security Standard (WSS)

Nicht relevant für den Wirkbetrieb der TI.

3.1.4 XML-Verschlüsselung – Symmetrisch

GS-A_4373 - XML-Verschlüsselung - symmetrisch
Alle Produkttypen, die XML-Dokumente mittels [XMLEnc-1.1] verschlüsseln, MÜSSEN die folgenden Vorgaben umsetzen:

- Als symmetrische Block-Chiffre muss AES [FIPS-197] mit einer Schlüssellänge von 256 Bit im Galois/Counter Mode (GCM) gemäß [NIST-SP-800-38D] mit der Tag-Länge von 128 Bit verwendet werden.
- Die IVs dürfen sich bei gleichem Schlüssel nicht wiederholen (vgl. [NIST-SP-800-38D#S.25] und [BSI-TR-02102-1#S. 24]). Der IV soll eine Bitlänge von 96 Bit

besitzen, seine Länge muss mindestens 96 Bit sein. Es wird empfohlen den IV zufällig zu wählen (vgl. [gemSpec_Krypt#GS-A_4367]).

- Hinweis: Im Normalfall ist davon auszugehen, dass für die Sicherung der Integrität und Authentizität der übertragenen Daten zudem noch eine Signatur der zu verschlüsselnden Daten notwendig ist.

[<=]

3.1.5 XML-Verschlüsselung – Hybrid

GS-A_4374 - XML-Verschlüsselung - Hybrid
Alle Produkttypen, die XML-Dokumente mittels [XMLEnc-1.1] hybrid verschlüsseln, MÜSSEN das XML-Dokument gemäß [gemSpec_Krypt#GS-A_4373] symmetrisch verschlüsseln, wobei der eingesetzte symmetrischer Schlüssel (jeweils) für eine spezifische Person oder Komponente asymmetrisch verschlüsselt wird.
(Hinweis: Analog zum Hinweis in [gemSpec_Krypt#GS-A_4373] gilt auch hier, dass im Normalfall für die Sicherung der Integrität und Authentizität der übertragenen Daten zudem noch eine Signatur dieser Daten notwendig ist.)
[<=]

GS-A_4375 - XML-Verschlüsselung - Hybrid, Schlüsseltransport
Alle Produkttypen, die XML-Dokumente mittels [XMLEnc-1.1] hybrid verschlüsseln, MÜSSEN für die Verschlüsselung des symmetrischen Schlüssel den Algorithmus RSAES-OAEP gemäß RFC 3447 [PKCS#1] oder Algorithmus RSAES-PKCS1-v1_5 unter Berücksichtigung von speziellen Maßnahmen gegen Seitenkanalangriffe (vgl. [BSI-TR-03116-1] S. 16) verwenden.
[<=]

GS-A_4376 - XML-Verschlüsselung - Hybrid, Schlüsseltransport RSAES-OAEP
Alle Produkttypen, die XML-Dokumente mittels [XMLEnc-1.1] hybrid verschlüsseln, SOLLEN für den Schlüsseltransport den Algorithmus RSAES-OAEP gemäß RFC 3447 [PKCS#1] verwenden.
[<=]

3.2 Karten-verifizierbare Authentifizierung und Verschlüsselung

3.2.1 Card-to-Card-Authentisierung G1

GS-A_4377 - Card-to-Card-Authentisierung G1
Alle Produkttypen, die die Card-to-Card-Authentisierung für Karten der Generation G1 durchführen, MÜSSEN dabei eine CV-Identität gemäß [gemSpec_Krypt#GS-A_4363] verwenden.
[<=]

Das Verfahren zur Durchführung der Card-to-Card-Authentisierung wird in [gemSpec_eGK_ObjSys] festgelegt.

3.2.2 Card-to-Server (C2S) Authentisierung und Trusted Channel G1

GS-A_4378 - Card-to-Server (C2S) Authentisierung und Trusted Channel G1
Alle Produkttypen, die die Card-to-Server-Authentisierung für Karten der Generation G1 durchführen, MÜSSEN die folgenden Vorgaben berücksichtigen:

- Die Authentisierung muss mit 3TDES analog [EN-14890-1#8.8] erfolgen und die Vorgaben der Tabelle Tab_KRYPT_011 berücksichtigen.
- Die Schlüsselvereinbarung muss analog zu [EN-14890-1#8.8.2] erfolgen.
- Das Verfahren zur Durchführung der Card-to-Server-Authentisierung erfolgt auf Grundlage von [EN-14890-1#8.8].

[<=]

Weitere Vorgaben finden sich in [gemSpec_SST_FD_VSDM].

C2S-Authentisierung bzw. der Trusted-Channel wird zwischen der eGK, dem zugeordneten CMS und dem zugeordneten VSDM-System verwendet.

Der Algorithmus 3TDES ist nach [BSI-TR-03116-1] in der TI nur noch im Rahmen der Kommunikation mit einer G1-Karte und nur noch bis Ende 2018 zulässig. Damit ist Ende 2018 eine obere Schranke für das Ende der G1-Karten.

Tabelle 13: Tab_KRYPT_011 Algorithmen für Card-to-Server-Authentifizierung

Algorithmen Typ	Algorithmus	Schlüssellänge
Authentifizierung und Verschlüsselung der Authentisierungsdaten	3TDES im CBC-Modus (OID 1.3.6.1.4.1.4929.1.8)	168 Bit zulässig bis Ende 2018

3.2.3 Card-to-Card-Authentisierung G2

GS-A_4379 - Card-to-Card-Authentisierung G2
Alle Produkttypen, die die Card-to-Card-Authentisierung für Karten der Generation G2 durchführen, MÜSSEN dabei eine CV-Identität gemäß [gemSpec_Krypt#GS-A_4365] verwenden.
[<=]

Das Verfahren zur Durchführung der Card-to-Card-Authentisierung wird in [gemSpec_COS] spezifiziert.

3.2.4 Card-to-Server (C2S) Authentisierung und Trusted Channel G2

GS-A_4380 - Card-to-Server (C2S) Authentisierung und Trusted Channel G2
Alle Produkttypen, die eine Card-to-Server-Authentisierung für Karten der Generation G2 durchführen, MÜSSEN die folgenden Vorgaben berücksichtigen:

- Die Authentisierung muss mit AES analog [EN-14890-1#8.8] erfolgen.
- Die Schlüsselvereinbarung muss analog zu [EN-14890-1#8.8.2] erfolgen.

[<=]

Das Verfahren zur Durchführung der Card-to-Server-Authentisierung wird in [gemSpec_COS] spezifiziert.

C2S-Authentisierung bzw. der Trusted-Channel wird zwischen der Karte und dem zugeordneten Management-System verwendet.

Der Algorithmus AES ist nach [BSI-TR-03116-1] in der TI bis Ende 2021+ (meint bis Ende des Betrachtungsraums der TR) zulässig.

GS-A_4381 - Schlüssellängen Algorithmus AES
Alle Produkttypen, die den Algorithmus AES nutzen, MÜSSEN die Schlüssellängen gemäß Tabelle Tab_KRYPT_012 nutzen.
[<=]

Tabelle 14: Tab_KRYPT_012 Algorithmen für Card-to-Server-Authentifizierung

Algorithmen Typ	Algorithmus	Schlüssellänge
Authentifizierung und Verschlüsselung der Authentisierungsdaten	AES im CBC-Modus (OID 2.16.840.1.101.3.4.1)	128 Bit zulässig bis Ende 2023+

3.3 Netzwerkprotokolle

Im Gegensatz zu kryptographischen Verfahren für den Integritätsschutz oder die Vertraulichkeit von Daten, bei denen keine direkte Kommunikation zwischen dem Sender bzw. dem Erzeuger und dem Empfänger stattfindet, kann bei Netzwerkprotokollen eine Aushandlung des kryptographischen Algorithmus erfolgen. Das Ziel der nachfolgenden Festlegungen ist es daher, jeweils genau einen verpflichtend zu unterstützenden Algorithmus festzulegen, so dass eine Einigung zumindest auf diesen Algorithmus immer möglich ist. Zusätzlich können aber auch optionale Algorithmen festgelegt werden, auf die sich Sender und Empfänger ebenfalls im Zuge der Aushandlung einigen können. Es darf jedoch durch keine der Komponenten vorausgesetzt werden, dass der Gegenpart diese optionalen Algorithmen unterstützt.

3.3.1 IPsec-Kontext

GS-A_4382 - IPsec-Kontext - Schlüsselvereinbarung
Alle Produkttypen, die die Authentifizierung, den Schlüsselaustausch und die verschlüsselte Kommunikation im IPsec-Kontext durchführen, MÜSSEN die Schlüsselvereinbarung mittels IKEv2 [RFC-7296] gemäß den folgenden Vorgaben durchführen:

- Zur Authentisierung MUSS eine Identität mit einem X.509-Zertifikat gemäß [gemSpec_Krypt#GS-A_4360] verwendet werden.
- Für „Hash und URL“ MUSS SHA-1 verwendet werden.
- Die Diffie-Hellman-Gruppe Gruppe 14 (definiert in [RFC-3526], verwendbar bis Ende 2023) MUSS für den Schlüsselaustausch unterstützt werden. Zusätzlich KÖNNEN Gruppen aus [BSI-TR-02102-3, Abschnitt 3.2.4, Tabelle 5], bei denen der Verwendungszeitraum ein „+“ enthält, verwendet werden.
- Der private DH-Exponent für den Schlüsselaustausch MUSS eine Länge von mindestens 256 Bit haben.
- Die Authentisierung der ephemeren (EC)DH-Parameter erfolgt durch eine Signatur der Parameter durch den jeweiligen Protokollteilnehmer. Bei dieser Signatur MUSS SHA-256 als Hashfunktion verwendet werden. Es SOLL die Authentisierungsmethode „Digital Signature“ nach [RFC-7427] dabei verwendet werden.

- Bei den symmetrische Verschlüsselungsalgorithmen MUSS AES mit 256 Bit Schlüssellänge im CBC-Modus unterstützt werden (sowohl für IKE-Nachrichten als auch später für die Verschlüsselung von ESP-Paketen). Es KÖNNEN weitere Verfahren nach [BSI-TR-02102-3, Abschnitt 3.2.1, Tabelle 2] bzw. [BSI-TR-02102-3, Abschnitt 3.3.1, Tabelle 7] verwendet werden.
- Für den Integritätsschutz (sowohl innerhalb von IKEv2 als auch anschließend für ESP-Pakete) MUSS HMAC mittels SHA-1 und SHA-256 (vgl. [gemSpec_Krypt#Hinweis-4382-1]) unterstützt werden. Es KÖNNEN weitere Verfahren nach [BSI-TR-02102-3, Abschnitt 3.2.3, Tabelle 4] bzw. [BSI-TR-02102-3, Abschnitt 3.3.1, Tabelle 8] verwendet werden.
- Als PRF MÜSSEN PRF_HMAC_SHA1 und PRF_HMAC_SHA2_256 (vgl. [gemSpec_Krypt#Hinweis-4382-1]) unterstützt werden. Es KÖNNEN weitere Verfahren nach [BSI-TR-02102-3, Abschnitt 3.2.2, Tabelle 3] verwendet werden.
- Schlüsselaktualisierung: die IKE-Lifetime darf maximal 24*7 Stunden betragen (Reauthentication). Die IPsec-SA-Lifetime darf maximal 24 Stunden betragen (Rekeying). Der Initiator soll nach Möglichkeit vor Ablauf der Lifetime das Rekeying anstoßen. Ansonsten muss der Responder bei Ablauf der Lifetime das Rekeying von sich aus sicherstellen, bzw. falls dies nicht möglich ist, die Verbindung beenden.
- Für die Schlüsselberechnung muss Forward Secrecy [BSI-TR-02102-1, S.ix] (in [RFC-7296] „Perfect Forward Secrecy“ genannt) gewährleistet werden. Meint die Wiederverwendung von zuvor schon verwendeten (EC-)Diffie-Hellman-Schlüsseln ([RFC-7296, Abschnitt 2.12]) ist nicht erlaubt.

[<=]

Hinweis-4382-1: In [NK-PP] wird mit FCS_COP.1/NK.HMAC und FCS_COP.1/NK.Hash die Unterstützung von SHA-1 und SHA-256 gefordert. Da für den Einsatz innerhalb einer HMAC-Funktion und innerhalb einer PRF die Einwegeigenschaft der Hashfunktion im Vordergrund steht und nicht die allgemeine Kollisionsresistenz, ist dort der Einsatz von SHA-1 noch zulässig (vgl. auch [BSI-TR-02102-3, Abschnitt 3.2.2, Tabelle 3, 4 und 8]). Es ist davon auszugehen, dass die Zulässigkeit von SHA-1 bei diesen beiden Einsatzzwecken zukünftig nicht mehr gegeben sein kann, und sowohl im NK als auch im VPN-Zugangsdienst, bspw. per Konfiguration, deaktiviert werden muss.

Ziel ist es zum Zeitpunkt der IKE-SA-Reauthentication ausgeführte Anwendungsfälle nicht zu unterbrechen. Aktuell wird aufgrund von TIP1-A_4492 im Rahmen der Reauthentication dem Konnektor eine neue (i.d.R. andere) VPN-TI-IP-Adresse zugewiesen, was dazu führt, dass bestehende TCP-Verbindungen in die TI effektiv zerstört und laufende Anwendungsfälle unterbrochen werden. Perspektivisch wird die folgende Anforderung als MUSS-Anforderung in TIP1-A_4492 integriert.

GS-A_5547 - gleiche VPN-IP-Adresse nach Reauthentication
Der VPN-Zugangsdienst KANN nach einer Reauthentication (vgl. GS-A_4382 Spiegelstrich „Schlüsselaktualisierung“) die gleiche VPN-IP-Adresse wie vor der Reauthentication vergeben. Die Reauthentication ist in Bezug auf TIP1-A_4492 nicht als „neue Verbindung/Neuaufbau des Tunnels“ zu betrachten.
[<=]

Da noch nicht alle VPN-Zugangsdienste technisch in der Lage sind GS-A_5547 umzusetzen werden als Symptomlinderung die Gültigkeitsdauern der ausgehandelten Schlüssel erhöht, auch in Anbetracht, dass weitere Sicherheitsmaßnahmen (bspw. TIP1-A_5389) umgesetzt werden neben den klassischen Prüfungen, die im Rahmen einer Reauthentication durchgeführt werden.

GS-A_5548 - Mindestgültigkeitszeiten IKE- und IPsec-SAs (Konnektor)

Der Konnektor MUSS die Konfiguration der Gültigkeitsdauern der IKE- bzw. IPsec-SAs auf (1) mindestens 90% und (2) kleiner als 100% der in GS-A_4382 Spiegelstrich „Schlüsselaktualisierung“ aufgeführten Maximalwerte setzen.
[<=]

Auszug Beispielkonfiguration /etc/ipsec.conf

```
ikelifetime=161h
lifetime=23h
margintime = 20m
rekeyfuzz = 40%
keyexchange=ikev2
```

GS-A_5549 - Mindestgültigkeitszeiten IKE- und IPsec-SAs (VPN-Zugangsdienst)
Der VPN-Zugangsdienst MUSS die Konfiguration der Gültigkeitsdauern der IKE- bzw. IPsec-SAs auf die in GS-A_4382 Spiegelstrich „Schlüsselaktualisierung“ aufgeführten Maximalwerte setzen.
[<=]

GS-A_5508 - IPsec make_before_break
Alle Produkttypen, die mittels IPsec Daten schützen, MÜSSEN die Reauthentication (vgl. [RFC-7296#2.8.3 „Reauthentication is done by […]“]) durchführen, indem die neue IKE-SA aufgebaut wird bevor die bestehende IKE-SA gelöscht wird.
[<=]

GS-A_4383 - IPsec-Kontext – Verschlüsselte Kommunikation
Alle Produkttypen, die mittels IPsec-Daten schützen, MÜSSEN dies ausschließlich auf Grundlage der in GS-A_4382 als zulässig aufgeführten Verfahren und Vorgaben tun.
[<=]

3.3.2 TLS-Verbindungen

GS-A_4385 - TLS-Verbindungen, Version 1.2
Alle Produkttypen, die Übertragungen mittels TLS durchführen, MÜSSEN die TLS-Version 1.2 [RFC-5246] unterstützen.
[<=]

Nach [RFC-5246, Abschnitt 7.4.1.2] muss ein TLS-Client beim Aufbau einer TLS-Verbindung (Handshake) die höchste von ihm unterstützte Version, also Version 1.2, als „favorite choice“ angeben. Mit [RFC-5246, Abschnitt 7.4.1.3] muss ein TLS-Server mit der höchsten von beiden Kommunikationspartnern unterstützten Version antworten, also nach GS-A_4385 Version 1.2. Damit wird zwischen Komponenten und Diensten, die GS-A_4385 umsetzen, nur noch die TLS-Version 1.2 verwendet.

Mittelfristig wird eine vollständige Migration auf TLS Version 1.2 angestrebt (vgl. auch [BSI-TR-02102-2, Abschnitt 3.2]), d. h. außer für den Konnektor und das KOM-LE-CM (s. u. GS-A_5530) wird die grundsätzliche Unterstützung von TLS-Version 1.1 freigestellt, und in einer späteren Migrationsphase wird diese Unterstützung (bzw. die Verwendung) untersagt.

GS-A_4386 - TLS-Verbindungen, optional Version 1.1
Alle Produkttypen, die Übertragungen mittels TLS durchführen, KÖNNEN die TLS-Version 1.1 [RFC-4346] unterstützen (oder auch nicht).
[<=]

Da alle aktuellen Webbrowser (vgl. Übersicht unter https://www.ssllabs.com/ssltest/clients.html und https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations) seit längerem TLS-Version 1.2 unterstützen ist eine Forderung der Unterstützung von TLS-Version 1.1 bei Diensten innerhalb der TI, die u. Um. von einem Primärsystem aus mittels eines Webbrowsers kontaktiert werden (bspw. VZD), nicht notwendig.

Komponenten, die direkt mit einem Primärsystem per TLS in Verbindung treten, sollen zunächst weiterhin die TLS-Version 1.1 unterstützen, um eine größtmögliche Interoperabilität zu erreichen.

GS-A_5530 - TLS-Verbindungen, Version 1.1
Der Konnektor und das KOM-LE-CM MÜSSEN die TLS-Version 1.1 unterstützen.**[<=]**

GS-A_4387 - TLS-Verbindungen, nicht Version 1.0
Alle Produkttypen, die Übertragungen mittels TLS durchführen, DÜRFEN NICHT die TLS-Version 1.0 unterstützen.**[<=]**

GS-A_5035 - Nichtverwendung des SSL-Protokolls
Alle Produkttypen, die Daten über Datenleitungen übertragen wollen, DÜRFEN NICHT das SSL-Protokoll unterstützen .**[<=]**

GS-A_4384 - TLS-Verbindungen
Alle Produkttypen, die Übertragungen mittels TLS durchführen, MÜSSEN die folgenden Vorgaben erfüllen:

- Zur Authentifizierung MUSS eine X.509-Identität gemäß [gemSpec_Krypt#GS-A_4359] verwendet werden.
- Als Cipher Suite MUSS TLS_DHE_RSA_WITH_AES_128_CBC_SHA oder TLS_DHE_RSA_WITH_AES_256_CBC_SHA verwendet werden.
- Es MUSS für die Schlüsselaushandlung Gruppe 14 (definiert in [RFC-3526], verwendbar bis Ende 2023) verwendet werden.
- Der private DH-Exponent für den Schlüsselaustausch MUSS eine Länge von mindestens 256 Bit haben.

[<=]

Für Embedded-Systeme (Konnektor, eHealth-KT) ist in diesem Zusammenhang lesenswert: [Oorschot-Wiener-1996].

Einen lesenswerten Abriss bekannter Angriffe auf TLS findet man in [TLS-Attacks], vgl. auch [Breaking-TLS].

GS-A_5541 - TLS-Verbindungen als TLS-Klient zur Störungsampel oder SM
Alle Produkttypen, die das TLS-Protokoll als TLS-Klient zur Störungsampel oder zum Service-Monitoring verwenden, KÖNNEN

(1) auf die explizite Prüfung der vom TLS-Server gesendeten DH-Gruppe und des öffentlichen DH-Schlüssels des TLS-Servers verzichten (vgl. GS-A_4384, Spiegelstrich 3), und

(2) davon ausgehen, dass der TLS-Server die Auswahl der TLS-Verbindungsparameter (TLS-Version, TLS-Ciphersuite etc.) korrekt, i.S.v. spezifikationskonform, durchführt.

[<=]

GS-A_5580 - TLS-Klient zur Störungsampel oder zum SM (Zertifikatsprüfung)
Alle Produkttypen, die das TLS-Protokoll als TLS-Klient zur Störungsampel oder zum Service Monitoring (die Störungsampel/ der Service-Monitoring-Server ist also TLS-

Server) verwenden, MÜSSEN das von der Störungsampel/SM präsentierte Zertifikat prüfen. Für diese Prüfung MUSS entweder TUC_PKI_018 oder die vereinfachte Zertifikatsprüfung (GS-A_5581 „TUC vereinfachte Zertifikatsprüfung“ (Komponenten-PKI)) verwendet werden.**[<=]**

Bei bestimmten Produkttypen, bspw. TSPs, beschränkt sich die Prüfung von Zertifikaten beim TLS-Verbindungsaufbau in Bezug auf die TI ausschließlich auf die Prüfung des Zertifikats der Störungsampel oder des Service Monitorings. Dafür ist der TUC_PKI_018 unangemessen leistungsstark und komplex. Deshalb wird folgend mit GS-A_5581 eine passgenauere Zertifikatsprüfung als Alternative definiert.

GS-A_5581 - "TUC vereinfachte Zertifikatsprüfung“ (Komponenten-PKI)
Alle Produkttypen, die eine Zertifikatsprüfung konform zu in dieser Anforderung definierten „TUC vereinfachte Zertifikatsprüfung“ durchführen wollen, erreichen dies indem sie folgende Vorgaben erfüllen.
(1) Es MUSS einen Prozess geben der authentisch und integer die Komponenten-CA-Zertifikate der TI regelmäßig (mindestens einmal pro Monat) ermittelt. Diese sind Basis für die folgenden Prüfschritte.
(2) Es MUSS geprüft werden, ob im vom TLS-Server präsentierten Zertifikat der korrekte (i. S. v. vom TLS-Client erwartete) FQDN enthalten ist (bspw. monitoring-update.stampel.telematik).
(3) Es MUSS geprüft werden, ob das präsentierte Zertifikat per Signaturprüfung rückführbar ist zu einem der CA-Zertifikate aus (1).
(4) Es MUSS geprüft werden, ob das präsentierte Zertifikat zeitlich gültig ist.

Wenn einer der Prüfschritte aus (2) bis (4) fehlschlägt, MUSS der Verbindungsaufbau abgebrochen werden.

Es gibt GS-A_5581 folgend in gemSpec_Krypt Anwendungshinweise.**[<=]**

Als Hilfestellung: für die Umsetzung von GS-A_5581 Spiegelstrich (1) kann man bspw. folgende Maßnahmen wählen.

(a) Übergabe bei einem Vororttermin in der gematik,

(b) Regelmäßiger Download über https://download.tsl.ti-dienste.de/

(c) Verwendung einer dedizierten Software zum Download, Signaturprüfung und Auswertung der TI-TSL (es existiert dafür jeweils mindestens eine Open-Source-Lösung und eine kommerzielle Lösung)

(d) oder andere Lösung, die die Integrität und Authentizität der Zertifikate sicherstellt.

Ziel ist es, dass für die Verbindung zur Störungsampel oder zum Service Monitoring auch einfach verfügbare und einfach verwendbare HTTPS-Clienten wie `wget` oder `curl` verwendet werden können.

Unter der Annahme, dass

(a) im Verzeichnis /etc/TI-Komponten-CAs die in GS-A_5581 Punkt (1) aufgeführten Zertifikate liegen und

(b) die an die Störungsampel zu sendende Information (i. d. R. unsignierte XML-Daten) in der Datei SOAP_Daten liegen,
erfüllen folgende Aufrufe die Punkt (2)-(4) aus GS-A_5581.
I.

```
wget --ca-directory=/etc/TI-Komponten-CAs --post-
```

```
file=SOAP_Daten https://monitoring-
update.stampel.telematik:8443/I_Monitoring_Message
```

II.

```
curl --capath /etc/TI-Komponten-CAs –d SOAP_Daten
https://monitoring-
update.stampel.telematik:8443/I_Monitoring_Message
```

GS-A_5542 - TLS-Verbindungen (fatal Alert bei Abbrüchen)
Alle Produkttypen, die das TLS-Protokoll verwenden, MÜSSEN sicherstellen, dass alle von ihnen durchgeführten Verbindungsabbrüche (egal ob im noch laufenden TLS-Handshake oder in einer schon etablierten TLS-Verbindung) mit einer im TLS-Protokoll aufgeführten Fehlermeldung (fataler Alert) angekündigt werden, außer das TLS-Protokoll untersagt dies explizit.
[<=**]**

Sicherheitsziel bei der Verwendung von TLS in der TI ist die Forward Secrecy [BSI-TR-02102-1, S. ix], was sich u. a. in den vorgegebenen CipherSuites (vgl. Tab_KRYPT_015 und Tab_KRYPT_016) widerspiegelt. Um dieses Ziel zu erreichen, muss sichergestellt werden, dass in regelmäßigen Abständen frisches Schlüsselmaterial über einen authentisierten Diffie-Hellman-Schlüsselaustausch gebildet wird, welches das alte Material ersetzt, wobei das alte Material sowohl im Klienten als auch im Server sicher gelöscht wird. Insbesondere bei der Nutzung von TLS-Resumption (vgl. [RFC-5246, S. 36] oder [RFC-5077]) kann die Dauer einer TLS-Session deutlich länger sein als die Lebensdauer der TCP-Verbindung innerhalb welcher der initiale Schlüsselaustausch stattgefunden hat. Aus diesem Grunde werden analog zu den IPsec-Vorgaben (vgl. [gemSpec_Krypt#GS-A_4383]) Vorgaben für die maximale Gültigkeitsdauer dieses Schlüsselmaterials gemacht (vgl. auch [SDH-2016]).

GS-A_5322 - Weitere Vorgaben für TLS-Verbindungen
Alle Produkttypen, die Übertragungen mittels TLS durchführen, MÜSSEN u. a. folgende Vorgaben erfüllen:

- Falls der Produkttyp als *Klient* oder als *Server* im Rahmen von TLS an einer Session-Resumption mittels SessionID (vgl. [RFC-5246, Abschnitt 7.4.1.2]) teilnimmt, MUSS er sicherstellen, dass nach spätestens 24 Stunden das über den Diffie-Hellman-Schlüsselaustausch ausgehandelte Schlüsselmaterial und alles davon abgeleitete Schlüsselmaterial (vgl. [RFC-5246, Abschnitt 8.1 und 6.3]) bei ihm sicher gelöscht wird.
- Falls der Produkttyp als *Klient* im Rahmen von TLS an einer Session-Resumption nach [RFC-5077] teilnimmt, MUSS er sicherstellen, dass nach spätestens 24 Stunden das über den Diffie-Hellman-Schlüsselaustausch ausgehandelte Schlüsselmaterial und alles davon abgeleitete Schlüsselmaterial (vgl. [RFC-5246, Abschnitt 8.1 und 6.3]) bei ihm sicher gelöscht wird. Damit verbundene SessionTickets MUSS er ebenfalls sicher löschen.
- Falls der Produkttyp als *Server* im Rahmen von TLS an einer Session-Resumption nach [RFC-5077] teilnimmt, MUSS er sicherstellen, dass nach spätestens 24 Stunden das über den Diffie-Hellman-Schlüsselaustausch ausgehandelte Schlüsselmaterial und alles davon abgeleitete Schlüsselmaterial (vgl. [RFC-5246, Abschnitt 8.1 und 6.3]) bei ihm sicher gelöscht wird. Damit verbundene SessionTickets MUSS er, falls bei ihm vorhanden, sicher löschen. Das Schlüsselmaterial, dass bei der Erzeugung des SessionTickets (für die Sicherung von Vertraulichkeit und Authentizität der SessionTickets) verwendet wird, MUSS spätestens alle 48 Stunden gewechselt werden und das alte Material

MUSS sicher gelöscht werden. Als kryptographische Verfahren zur Erzeugung/Sicherung der SessionTickets MÜSSEN ausschließlich nach [BSI-TR-03116-1] zulässige Verfahren verwendet werden und das Schlüsselmaterial muss die Entropieanforderungen gemäß [gemSpec_Krypt#GS-A_4368] erfüllen.

- Falls ein Produkttyp als *Klient* oder *Server* im Rahmen von TLS die Renegotiation unterstützt, so MUSS er dies ausschließlich nach [RFC-5746] tun. Ansonsten MUSS er die Renegotiation-Anfrage des Kommunikationspartners ablehnen.

[<=]

Aktuell gibt es in der TI keine Anwendungsfälle (Wechsel der kryptographischen Identität innerhalb einer TLS-Verbindung oder erzwungene Schlüssel-„Auffrischung“ der Sitzungsschlüssel), die eine Session-Renegotiation im Rahmen von TLS unmittelbar erforderlich machen. Lesenswert bez. des Themas Sicherheitsprobleme mit TLS-Session-Renegotiation ist [IR-2014, S.181ff] und allgemein [CM-2014].

Es hat sich gezeigt, dass es notwendig ist weitere Vorgaben zur TLS-Renegotiation für die Sicherstellung der Interoperabilität zwischen Komponenten und Diensten zu machen.

GS-A_5524 - TLS-Renegotiation eHealth-KT
Das eHealth-KT MUSS beim einen TLS-Verbindungsaufbau die TLS-Extension „renegotiation_info“ gemäß [RFC-5746] senden, unabhängig davon ob das eHealth-KT TLS-Renegotiation unterstützt oder nicht unterstützt. Im weiteren TLS-Protokollverlauf MUSS das eHealth-KT eines der beiden folgenden Verhalten aufweisen:

1. Entweder das eHealth-KT lehnt jede Renegotiation mit einem „no_renegotiation“-Alert ab, oder
2. das eHealth-KT unterstützt die Renegotiation gemäß [RFC-5746], wobei ausschließlich „Secure Renegotiation“ durch das eHealth-KT akzeptiert werden (d.h., falls das „secure_renegotiation“-flag [RFC-5746#3.7] gleich FALSE ist, muss das KT die Renegotiation mit einem „no_renegotiation“-Alert ablehnen).

[<=]

GS-A_5525 - TLS-Renegotiation Konnektor
Der Konnektor MUSS den RFC 5746 (TLS-Renegotiation-Indication-Extension [RFC-5746]) unterstützen und nur „Secure Renegotiation“ erlauben und durchführen.
[<=]

Für eine Java-Implementierung bedeutet dies, dass allowLegacyHelloMessages und allowUnsafeRenegotiation jeweils auf false gesetzt sind ("Modus Strict", http://www.oracle.com/technetwork/java/javase/overview/tlsreadme2-176330.html).

Da der Angriff [Ray-2009], der zur Erstellung des [RFC-5746] führte, praktisch durchführbar war, wurde die Mehrzahl der existierenden TLS-Bibliotheken relativ zügig angepasst (Timeline in [IR-2014, S. 190, Abbildung 7.2]). (Vgl. die erste Spalte „Secure Renegotiation“ bei https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations#Extensions) Um für den unwahrscheinlichen Fall, dass aktuell ein schon bestehender Fachdienst Probleme bei der Umsetzung der folgenden Anforderung hat, wurde diese als SOLL-Anforderung formuliert. Es ist geplant diese Anforderung zukünftig in eine MUSS-Anforderung zu ändern.

GS-A_5526 - TLS-Renegotiation-Indication-Extension
Alle Produkttypen, die das TLS-Protokoll verwenden, SOLLEN den RFC 5746 (TLS-Renegotiation-Indication-Extension [RFC-5746]) unterstützen.
[<=]

Die folgende Anforderung hat den Zweck die Interoperabilität zwischen Konnektor und Intermediär sicherzustellen.

GS-A_5527 - TLS-Renegotiation-Indication-Extension Intermediär
Der Intermediär MUSS den RFC 5746 (TLS-Renegotiation-Indication-Extension [RFC-5746]) unterstützen und nur „Secure Renegotiation“ erlauben und durchführen.
[<=]

Für eine verbesserte Interoperabilität zu bestimmten TLS-Implementierungen (bspw. SChannel, vgl. auch(https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations bzw. https://www.ssllabs.com/ssltest/clients.html) sollen im Konnektor zusätzlich zu den Ciphersuiten aus GS-A_4384 weitere Ciphersuiten unterstützt werden. Mit der mittelfristigen Anhebung des zu erreichenden Sicherheitsniveaus auf 120 Bit (vgl. [SOG-IS-2018] und [BSI-TR-03116-1]) werden die folgenden Ciphersuiten mittelfristig verpflichtend. In diesem Kontext spielt die Performanz (3000 Bit Diffie-Hellman vs. 256 Bit Elliptic Curve Diffie-Hellman) bei Embedded-Geräten wie dem Konnektor eine wichtige Rolle.

GS-A_5345 - TLS-Verbindungen Konnektor
Der Konnektor MUSS für die TLS gesicherten Verbindungen neben den in [gemSpec_Krypt#GS-A_4384] aufgeführten Ciphersuiten folgende Vorgaben umsetzen:

1. Der Konnektor MUSS zusätzlich folgende Ciphersuiten unterstützen:
   - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC0, 0x13),
   - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC0, 0x14),
   - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC0, 0x27),
   - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC0, 0x28),
   - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2f)und
   - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x30).
2. Der Konnektor KANN weitere Ciphersuiten aus [TR-02102-2, Abschnitt 3.3.1 Tabelle 1] unterstützen.
3. Falls Ciphersuiten aus Spiegelstrich (1) oder (2) unterstützt werden,
   a. MÜSSEN bei dem ephemeren Elliptic-Curve-Diffie-Hellman-Schlüsselaustausch die Kurven P-256 oder P-384 [FIPS-186-4] unterstützt werden,
   b. MÜSSEN die Kurven brainpoolP256r1 und brainpoolP384r1 (vgl. [RFC-5639] und [RFC-7027]) unterstützt werden.

   Andere Kurven SOLLEN NICHT verwendet werden.
4. Falls Ciphersuiten aus (1) oder (2) unterstützt werden, so MÜSSEN diese im CC-Zertifizierungsverfahren berücksichtigt werden.

[<=]

Von einem TLS-Server, dessen Kommunikationspartner Standard-Webbrowser sind (bspw. einem Webserver), wird wie folgt eine Webbrowser-Interoperabilität bez. der unterstützten TLS-Ciphersuiten gefordert.

GS-A_5339 - TLS-Verbindungen, erweiterte Webbrowser-Interoperabilität
Alle Produkttypen, die TLS verwenden und bei denen insbesondere Webbrowser-Interoperabilität (Webportale, Download-Punkte o. Ä.) wichtig ist, MÜSSEN zur Absicherung der TLS-Übertragung neben der in [gemSpec_Krypt#GS-A_4384] aufgeführten Vorgaben zusätzlich Folgendes sicherstellen:

1. Der Produkttyp MUSS zusätzlich folgende Ciphersuiten unterstützen:
   - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC0, 0x14),
   - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC0, 0x13),
   - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x30) und
   - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2F).
2. Der TLS-Server KANN weitere Cipher-Suiten aus [TR-02102-2, Abschnitt 3.3.1 Tabelle 1] unterstützen.
3. Bei dem ephemeren Elliptic-Curve-Diffie-Hellman-Schlüsselaustausch MÜSSEN die Kurven P-256 oder P-384 [FIPS-186-4] unterstützt werden. Daneben KÖNNEN die Kurven brainpoolP256r1, brainpoolP384r1 oder brainpoolP512r1 (vgl. [RFC-5639] und [RFC-7027]) unterstützt werden.
   Andere Kurven SOLLEN NICHT verwendet werden (Hinweis: die Intention des letzten Satzes ist insbesondere, dass die Ordnung des Basispunktes in E(F_p) nicht zu klein werden darf).

[<=]

Hinweis: hinter den folgenden Identifier-n verbirgt sich kryptographisch gesehen jeweils die gleiche Kurve:

ansix9p256r1	**[ANSI-X9.62#L.6.4.3]**
ansip256r1	http://oid-info.com/get/1.2.840.10045.3.1.7
prime256v1	[RFC-3279], `openssl ecparam -list_curves`
secp256r1	[RFC-5480], http://www.secg.org/collateral/sec2_final.pdf
P-256	[FIPS186-4]

Analog P-384 [FIPS186-4]:

ansix9p384r1	**[ANSI-X9.62#L.6.5.2]**
ansip384r1	http://oid-info.com/get/1.3.132.0.34
prime384v1	[RFC-3279], `openssl ecparam -list_curves`
secp384r1	[RFC-5480], http://www.secg.org/collateral/sec2_final.pdf
P-384	[FIPS186-4]

Der VZD wird u. Um. direkt von einem Webbrowser angesprochen, daher wird für eine größere Interoperabilität zu verschiedenen Webbrowsern von ihm die Unterstützung zusätzlicher TLS-Ciphersuiten gefordert.

GS-A_5482 - zusätzliche TLS-Ciphersuiten für VZD
Der VZD MUSS in Bezug auf TLS neben den in [gemSpec_Krypt#GS-A_4384] aufgeführten Ciphersuiten folgende Vorgaben umsetzen:

1. Der VZD MUSS zusätzlich folgende Ciphersuiten unterstützen:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xC0, 0x13),
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xC0, 0x14),
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xC0, 0x27),
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xC0, 0x28),
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2f)und
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x30).

2. Der VZD KANN weitere Ciphersuiten aus [TR-02102-2, Abschnitt 3.3.1 Tabelle 1] unterstützen.
3. Der VZD MUSS bei den TLS-Ciphersuiten aus Spiegelstrich (1) oder (2) bei dem ephemeren Elliptic-Curve-Diffie-Hellman-Schlüsselaustausch die Kurven P-256 oder P-384 [FIPS-186-4] unterstützen. DanebenKÖNNEN die Kurven brainpoolP256r1 und brainpoolP384r1 (vgl. [RFC-5639] und [RFC-7027]) unterstützt werden. Andere Kurven SOLLEN NICHT verwendet werden (Hinweis: die Intention des letzten Satzes ist insbesondere, dass die Ordnung des Basispunktes in E(F_p) nicht zu klein werden darf).

[<=]

3.3.3 DNSSEC-Kontext

GS-A_4388 - DNSSEC-Kontext
Alle Produkttypen, die DNSSEC verwenden, MÜSSEN die Algorithmen und Vorgaben gemäß Tabelle Tab_KRYPT_017 erfüllen.
[<=]

Tabelle 15 Tab_KRYPT_017 Algorithmen für DNSSEC

Algorithmen Typ	Algorithmus	Schlüssellänge
TSIG – symmetrischer Schlüssel zur Absicherung der Transaktionskanäle zwischen zwei Name-Server-Instanzen bei Zonentransfers, Änderungsbenachrichtigungen, dynamischen Updates und rekursiven Queries.	HMAC-SHA-256	256 Bit
DNSSEC ZSK Asymmetrische Schlüssel zur Wahrung der Authentizität und Integrität von Zonendatenobjekten.	RSA-SHA-256 [RFC-5702]	2048 Bit
DNSSEC KSK Asymmetrische Schlüssel zur Wahrung der Authentizität und Integrität von Zonendatenobjekten.	RSA-SHA-256 [RFC-5702]	2048 Bit

Hinweis: Nach [RFC-5702] ist die Verwendung von SHA-256 [FIPS-180-4] möglich. Schlüssellängen von RSA zwischen 512 bis 4096 Bit sind seit den Anfängen von DNSSEC möglich. Bei TSIG ist nach [RFC-4635] auch SHA-256 verwendbar und bspw. von bind seit der Version 9.5 unterstützt.

3.4 Masterkey-Verfahren (informativ)

Die gematik wurde aufgefordert, beispielhaft ein mögliches Ableitungsverfahren für einen versichertenindividuellen symmetrischen Schlüssel auf Grundlage eines Ableitungsschlüssels (Masterkey) aufzuführen. Ein Kartenherausgeber ist frei in der Wahl seines Ableitungsverfahrens. Jedoch müssen beim Einsatz eines Ableitungsverfahrens, um die Qualität der Ableitung zu garantieren, insbesondere folgende Punkte beachtet werden:

- Der Ableitungsprozess muss unumkehrbar und nicht-vorhersehbar sein, um sicherzustellen, dass die Kompromittierung eines abgeleiteten Schlüssels nicht den Ableitungsschlüssel oder andere abgeleitete Schlüssel kompromittiert.
- Bei einer Schlüsselableitung (im Sinne von [ISO-11770]) basiert die kryptographische Stärke der abgeleiteten Schlüssel auf der Ableitungsfunktion und der kryptographischen Stärke des geheimen Ableitungsschlüssels (insbesondere hier dessen Entropie). Die Entropie der abgeleiteten Schlüssel ist kleiner gleich der Entropie des geheimen Ableitungsschlüssels. Um die Entropie der abgeleiteten Schlüssel sicherzustellen, muss die Entropie des geheimen Ableitungsschlüssels (deutlich) größer sein als die zu erreichende Entropie der abgeleiteten Schlüssel.
- Der Betreiber eines Schlüsseldienstes muss im Falle des Einsatzes einer Schlüsselableitung (nach [ISO-11770]) in seinem Sicherheitskonzept Maßnahmen für das Bekanntwerden von Schwächen des kryptographischen Verfahrens, welche die Grundlage der Schlüsselableitung ist, darlegen.

Ein Kartenherausgeber hat auch die Freiheit, gar kein Ableitungsverfahren zu verwenden, sondern alle symmetrischen SK.CMS aller seiner Karten sicher in seinem RZ vorzuhalten.

Ziel des Masterkey-Verfahrens zur Ableitung eines versichertenindividuellen Schlüssels ist es, aus einem geheimen Masterkey und einem öffentlichen versichertenindividuellen Merkmal einen geheimen symmetrischen Schlüssel abzuleiten, der zur Absicherung der Verbindung zwischen CMS und Smartcard verwendet wird. Öffentlich bedeutet an dieser Stelle nicht, dass die Merkmale selbst nicht schützenswert sind, es soll jedoch ausdrücken, dass die Vertraulichkeit des versichertenindividuellen Schlüssels nicht von der Geheimhaltung dieser Merkmale abhängt. Die Vertraulichkeit der Daten muss durch die Geheimhaltung des Masterkeys gewährleistet sein. Das bedeutet, die Geheimhaltung anderer Daten als des Masterkeys darf für die Vertraulichkeit der Daten nicht notwendig sein. Die Durchführung dieses Verfahrens muss bei gleichen Eingangsparametern immer das gleiche Ergebnis generieren.

Für die Durchführung des Algorithmus wird neben dem Masterkey auch noch mindestens ein versichertenindividuelles Merkmal verwendet. Die Auswahl des Merkmals ist fachlich motiviert und wird daher in diesem Dokument nicht spezifiziert. Das in Tabelle 20 beispielhafte Verfahren besteht aus einer Kombination von AES-Verschlüsselung [FIPS-197] und Hashwert-Bildung. Die Schlüssel- bzw. Hashwert-Länge ergibt sich gemäß Tabelle 21.

Tabelle 16: Tab_KRYPT_018 Ablauf zur Berechnung eines versichertenindividuellen Schlüssels

Reihenfolge	Beschreibung	Formale Darstellung

1	Bildung eines Hashwertes über dem versichertenindividuellen Merkmal unter Verwendung eines statischen Padding-Verfahrens für den Fall, dass das versichertenindividuelle Merkmal in seiner Länge nicht der Blocklänge des Hash-Algorithmus entspricht. Im Ergebnis wird ein versichertenindividuelles Merkmal geeigneter Länge für den nächsten Schritt erzeugt.	HASH#1 = SHA-256(versichertenindividuelles Merkmal)
2	AES-Verschlüsselung des Resultats mit dem Masterkey. Durch die Verschlüsselung an dieser Stelle ist sichergestellt, dass der versichertenindividuelle Schlüssel nur durch den Besitzer des geheimen Masterkeys erzeugt werden kann.	ENC#1 = AES-256(HASH#1)
3	Bildung eines Hashwertes über dem Ergebnis des vorherigen Verarbeitungsschritts. Dies stellt sicher, dass ein Schlüssel geeigneter Länge erzeugt wird.	Versichertenindividueller Schlüssel = SHA-256(ENC#1)

In der nachfolgenden Tabelle werden Kürzel entsprechend der Definition aus Abschnitt 3.2.3 verwendet.

Tabelle 17: Tab_KRYPT_019 eingesetzte Algorithmen für die Ableitung eines versichertenindividuellen Schlüssels

Algorithmen Typ	Algorithmus	Unterverfahren
Masterkey-Verfahren für die Generierung des versichertenindividuellen Schlüssel innerhalb eines CMS	AES basiertes Verfahren gemäß vorheriger Definition	AES-256 SHA-256 anwendbar bis Ende 2023+

3.5 Hybride Verschlüsselung binärer Daten

Für die hybride Verschlüsselung werden die Daten zunächst symmetrisch mittels eines zufällig gewählten geheimen symmetrischen Schlüssels verschlüsselt. Der geheime Schlüssel wird im Anschluss asymmetrisch für jeden Empfänger separat verschlüsselt.

Hinweis: unter binären Daten sind im gesamten Dokument beliebige Daten insbesondere beliebigen Typs (Text, HTML, PDF, JPG etc.) zu verstehen. Es gilt das Prinzip: das Spezielle vor dem Allgemeinen: gibt es weitere spezielle Vorgaben für bestimmte Datenformate, sind diese für die entsprechenden Daten verpflichtend (überschreiben oder ergänzen die allgemeinen Vorgaben).

3.5.1 Symmetrischer Anteil der hybriden Verschlüsselung binärer Daten

GS-A_4389 - Symmetrischer Anteil der hybriden Verschlüsselung binärer Daten

Produkttypen, die die hybride Verschlüsselung binärer Daten durchführen, MÜSSEN für den symmetrischen Anteil der Verschlüsselung die folgenden Vorgaben berücksichtigen:

- Als symmetrische Block-Chiffre muss AES [FIPS-197] mit einer Schlüssellänge von 256 Bit im Galois/Counter Mode (GCM) gemäß [NIST-SP-800-38D] mit der Tag-Länge von 128 Bit verwendet werden.
- Die IVs dürfen sich bei gleichem Schlüssel nicht wiederholen (vgl. [NIST-SP-800-38D#S.25] und [BSI-TR-02102-1#S.24]). Der IV soll eine Bitlänge von 96 Bit besitzen, seine Länge muss mindestens 96 Bit sein. Es wird empfohlen den IV zufällig zu wählen (vgl. [gemSpec_Krypt#GS-A_4367]).
- Hinweis: Im Normalfall ist davon auszugehen, dass für die Sicherung der Integrität und Authentizität der zu verschlüsselnden Daten zudem noch eine Signatur dieser Daten notwendig ist.

[<=]

Hinweis: In [RFC-5084] findet man Informationen über die Verwendung von AES-GCM innerhalb von CMS [RFC-5652].

3.5.2 Asymmetrischer Anteil der hybriden Verschlüsselung binärer Daten

GS-A_4390 - Asymmetrischer Anteil der hybriden Verschlüsselung binärer Daten
Produkttypen, die die hybride Verschlüsselung binärer Daten durchführen, MÜSSEN für den asymmetrischen Anteil der Verschlüsselung die folgenden Vorgaben berücksichtigen:

- Als asymmetrisches Verschlüsselungsverfahren soll RSAES-OAEP gemäß [PKCS#1, Kapitel 7.1] verwendet werden.
- Sofern eine Implementierung der Systeme mit RSAES-OAEP nicht möglich ist, muss RSAES-PKCS1-v1-5 gemäß [PKCS#1 Kapitel 7.2] verwendet werden. Die Gültigkeit dieses Verfahrens ist bis Ende 2017 beschränkt. Bei der Verwendung dieses Verfahrens ist besonders auf die zusätzliche Sicherung der Integrität und Authentizität der verschlüsselten Daten zu achten, da Angriffe bekannt sind bei denen ein Angreifer korrekt dekodierbare Chiffretexte erzeugen kann.
- Als Mask-Generation-Function für die Verwendung in RSAES-OAEP muss MGF 1 mit SHA-256 als Hash-Funktion gemäß [PKCS#1, Anhang B.2.1] verwendet werden.

[<=]

3.6 Symmetrische Verschlüsselung binärer Daten

GS-A_5016 - Symmetrische Verschlüsselung binärer Daten
Produkttypen, die die symmetrische Verschlüsselung binärer Daten durchführen, MÜSSEN die folgenden Vorgaben berücksichtigen:

- Als symmetrische Block-Chiffre muss AES [FIPS-197] mit einer Schlüssellänge von 256 Bit im Galois/Counter Mode (GCM) gemäß [NIST-SP-800-38D] mit der Tag-Länge von 128 Bit verwendet werden.
- Die IVs dürfen sich bei gleichem Schlüssel nicht wiederholen (vgl. [NIST-SP-800-38D#S.25] und [BSI-TR-02102-1#S.24]). Der IV soll eine Bitlänge von 96 Bit besitzen, seine Länge muss mindestens 96 Bit sein. Es wird empfohlen den IV zufällig zu wählen (vgl. [gemSpec_Krypt#GS-A_4367]).

- Hinweis: Im Normalfall ist davon auszugehen, dass für die Sicherung der Integrität und Authentizität der übertragenen Daten zudem noch eine Signatur der zu verschlüsselnden Daten notwendig ist.

[<=]

Hinweis: In [RFC-5084] findet man Informationen über die Verwendung von AES-GCM innerhalb von CMS [RFC-5652].

3.7 Signatur binärer Inhaltsdaten (Dokumente)

GS-A_5080 - Signaturen binärer Daten (Dokumente)
Alle Produkttypen, die CMS-Signaturen [RFC-5652] von Inhaltsdaten (wie bspw. Textdokumenten ungleich PDF/A) erzeugen oder prüfen, MÜSSEN die Algorithmen und Vorgaben der Tabelle Tab_KRYPT_020 erfüllen.
[<=]

Tabelle 18: Tab_KRYPT_020 Algorithmen für die Erzeugung und Prüfung von binären Daten im Kontext von Dokumentensignaturen

Signaturbestandteil	Beschreibung	Algorithmus	Anmerkung
Signaturstandard	Signaturstandard	ETSI TS 101 733 V1.7.4 (2008-07) Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES) [ETSI-CAdES]	Die Verwendung des Standards ist für die Signatur von Dokumenten verpflichtend die mittels CMS (PKCS#7) erzeugt werden.
kryptographisches Signaturverfahren	Algorithmus für die Berechnung des Nachrichten Digest und die Verschlüsselung mit dem privaten Schlüssel	**RSASSA-PSS mit SHA256** bis nach Ende 2023+ verwendbar (Ende des Betrachtungshorizonts)	Die Verwendung einer dieser Algorithmen ist verpflichtend. Alle hier aufgeführten Signaturverfahren müssen von einer Signaturprüfenden Komponente überprüfbar sein.
DigestMethod	Methode zur Berechnung eines Digest der zu signierenden Bereiche	SHA-256	Die Verwendung des Algorithmus ist verpflichtend.
Kryptographisches Token	Kryptographisches Token für die Signatur, bestehend aus einem privaten	Identitäten gemäß einem der folgenden Abschnitte 2.1.1.1	Die Auswahl des kryptographischen Tokens ist von dem jeweiligen

	Schlüssel und einem zugehörigen X.509-Zertifikat	2.1.1.2	Einsatzzweck abhängig.

3.8 Signaturen innerhalb von PDF/A-Dokumenten

GS-A_5081 - Signaturen von PDF/A-Dokumenten
Alle Produkttypen, die in PDF/A-Dokumenten [PDF/A-2] Signaturen einbetten/erzeugen oder diese Signaturen prüfen, MÜSSEN die Algorithmen und Vorgaben der Tabelle Tab_KRYPT_021 erfüllen.
[<=]

Tabelle 19: Tab_KRYPT_021 Algorithmen für die Erzeugung und Prüfung von PDF/A-Dokumentensignaturen

Signaturbestandteil	Beschreibung	Algorithmus	Anmerkung
Signaturstandard	Signaturstandard	ETSI TS 102 778-3 V1.2.1, PDF Advanced Electronic Signature Profiles; Part 3: PAdES Enhanced – PAdES-BES and PAdES-EPES Profiles Technical Specification, 2010 [PAdES-3]	Die Verwendung des Standards ist für die Signatur von PDF/A [PDF/A-2] Dokumenten verpflichtend, die mittels eingebetteter Signaturen signiert werden.
kryptographisches Signaturverfahren	Algorithmus für die Berechnung des Nachrichten Digest und die Verschlüsselung mit dem privaten Schlüssel	**RSASSA-PSS mit SHA256** bis nach Ende 2023+ verwendbar (Ende des Betrachtungshorizonts)	Die Verwendung einer dieser Algorithmen ist verpflichtend. Alle hier aufgeführten Signaturverfahren müssen von einer Signaturprüfenden Komponente überprüfbar sein.
DigestMethod	Methode zur Berechnung eines Digest der zu signierenden Bereiche	**SHA-256**	Die Verwendung des Algorithmus ist verpflichtend.
Kryptographisches Token	Kryptographisches Token für die Signatur, bestehend aus einem privaten	Identitäten gemäß einem der folgenden Abschnitte 2.1.1.1	Die Auswahl des kryptographischen Tokens ist von dem jeweiligen

	Schlüssel und einem zugehörigen X.509-Zertifikat	2.1.1.2	Einsatzzweck abhängig.

3.9 Kartenpersonalisierung

Vgl. auch Abschnitt 2.4 (Schlüsselerzeugung).

GS-A_4391 - MAC im Rahmen der Personalisierung der eGK
Der Herausgeber der eGK MUSS sicherstellen, dass bei der Personalisierung der eGK die Daten bei der Übermittlung integritätsgeschützt werden. Für die Absicherung der Integrität ist in diesem Kontext der AES-256 CMAC nach [NIST-SP-800-38B] (vgl. [BSI-TR-03116-1#3.2.2, 4.5.2]) zu verwenden.
Die Länge des CMAC muss 128 Bit betragen.
Nach [NIST-SP-800-38B#S.13] sollen nicht mehr als 2^{48} Nachrichtenblöcke (2^{22} GByte) mit demselben Schlüssel verarbeitet werden. Nach [NIST-SP-800-38B#S.14] ist ein CMAC anfällig für Replay-Attacken, was bei der Anwendung des CMACs zu berücksichtigen ist.
[<=]

3.10 Bildung der pseudonymisierten Versichertenidentität

GS-A_4392 - Algorithmus im Rahmen der Bildung der pseudonymisierten Versichertenidentität
Alle Produkttypen, die pseudonymisierte Versichertenidentitäten berechnen, MÜSSEN den Hash-Algorithmus SHA-256 [FIPS-180-4] verwenden.**[<=]**

3.11 Spezielle Anwendungen von Hashfunktionen

GS-A_4393 - Algorithmus bei der Erstellung von Hashwerten von Zertifikaten oder öffentlichen Schlüsseln
Alle Produkttypen, die Fingerprints eines öffentlichen Schlüssels oder eines Zertifikates erstellen, MÜSSEN den Hash-Algorithmus SHA-256 [FIPS-180-4] dafür verwenden.**[<=]**

Erläuterung:

Alle CAs und der TSL-Dienst müssen im Rahmen ihrer Prozesse öffentliche Schlüssel oder Zertifikate (bspw. auf Webseiten) veröffentlichen. Dabei wird auch jeweils der SHA-256 Hashwert mit veröffentlicht.

Hersteller einer gSMC-KT müssen den Hashwert des auf der Karte befindlichen Zertifikats in MF/DF.KT/EF.C.SMKT.AUT.R2048 entweder auf dem ID-1-Kartenkörper drucken (das ID-000-Modul ist dann herausbrechbar) oder ausgedruckt mitliefern. Der Konnektor muss den Hashwert des Zertifikats bei initialen Pairing mit dem KT berechnen und dem Administrator präsentieren.

Innerhalb der CertHash-Extension als Teil einer OCSP-Response wird vom TSP ein SHA-256 Hashwert des Zertifikats, über das eine Sperrinformation gegeben wird, mitgeliefert.

GS-A_5131 - Hash-Algorithmus bei OCSP/CertID

Alle Produkttypen, die OCSP-Anfragen stellen oder beantworten, MÜSSEN bei der Erstellung und Verwendung der CertID-Struktur (vgl. [RFC-6960, Abschnitt 4.1.1] oder [RFC-2560, Abschnitt 4.1.1]) den Hash-Algorithmus SHA-1 [FIPS-180-4] verwenden. Ein OCSP-Server KANN auch zusätzlich andere Hashfunktionen im Rahmen der CertID, die nach [BSI-TR-03116-1] zulässig sind, unterstützen.
[<=**]**

3.11.1 Hashfunktionen und OCSP (informativ)

Es hat sich gezeigt, dass zum folgenden Themenkomplex eine Erläuterung hilfreich ist.

Im Zusammenspiel OCSP-Anfrage und OCSP-Antwort werden an drei Stellen Hashfunktionen verwendet, die theoretisch alle paarweise verschieden sein können.

Erste Stelle: Zunächst erzeugt ein OCSP-Client eine OCSP-Anfrage (vgl. [RFC-6960, Abschnitt 4.1.1] oder [RFC-2560, Abschnitt 4.1.1]). Dafür muss dieser u. a. eine CertID-Datenstruktur erzeugen:

```
CertID          ::=     SEQUENCE {
    hashAlgorithm       AlgorithmIdentifier,
    issuerNameHash      OCTET STRING, -- Hash of issuer's DN
    issuerKeyHash       OCTET STRING, -- Hash of issuer's public key
    serialNumber        CertificateSerialNumber }
```

Bei der Wahl der Hashfunktion kann er sich nur darauf verlassen, dass der OCSP-Responder als Hashalgorithmus (vgl. „hashAlgorithm“-Datenfeld) SHA-1 [FIPS-180-4] unterstützt. Für den Anfragenden und den OCSP-Responder gilt dementsprechend GS-A_5131. Er muss SHA-1 für die CertID-Struktur verwenden. Ein OCSP-Responder, der zusätzlich weitere Hashfunktionen unterstützt, muss nichts zurückbauen – er darf auch so in der TI arbeiten.

Warum ist der Einsatz von SHA-1 an dieser Stelle kryptographisch gesehen ausreichend? Da (1) ein OCSP-Responder der TI nicht für beliebige CAs arbeitet (Wahl von DN und öffentlichen Schlüssel ist damit beschränkt) und (2) i. d. R. die CertHash-Extension Teil der OCSP-Antwort ist und innerhalb der CertHash-Extension in der TI eine kryptographisch hochwertigen Hashfunktion verwendet wird, ist die Verwendung von SHA-1 hier aus Sicherheitssicht betrachtet unbedenklich. (Vgl. analoges Vorgehen BNetzA-OCSP-Responder für den qualifizierten Vertrauensraum.) Es ist also sichergestellt, dass zwischen OCSP-Client und -Responder keine (evtl. von einem Angreifer böswillig herbeigeführten) Unklarheiten darüber entstehen können über welches Zertifikat gerade gesprochen wird. Es geht bei GS-A_5131 vornehmlich um die Interoperabilität von OCSP-Client und OCSP-Responder.

Die optionale Signatur einer OCSP-Anfrage wird in der TI nicht verwendet, damit ist die dort verwendete Hashfunktion die aktuelle Betrachtung irrelevant.

Zweite Stelle: Für die Beantwortung der OCSP-Anfrage erzeugt der OCSP-Responder u. a. eine CertHash-Datenstruktur:

```
id-commonpki-at-certHash OBJECT IDENTIFIER ::= {1 3 36 8 313}
CertHash ::= SEQUENCE {
     hashAlgorithm   AlgorithmIdentifier,  -- The identifier
     -- of the algorithm that has been used the hash value below.
     certificateHash OCTET STRING }
```

Hierfür muss eine kryptographisch hochwertige (nach [BSI-TR-03116-1] zulässige) Hashfunktion verwendet werden. Normativ ist an dieser Stelle: „GS-A_4393 Algorithmus bei der Erstellung von Hashwerten von Zertifikaten oder öffentlichen Schlüsseln“.

Spätestens an dieser Stelle können OCSP-Client und OCSP-Server sich sicher sein, ob sie über das gleiche Zertifikat sprechen.

Dritte Stelle: Die OCSP-Response muss am Ende vom OCSP-Responder signiert werden. Dafür ist die Vorgabe aus Tab_KRYPT_002 „Signatur der OCSP-Response“ normativ, welche über die für die jeweiligen Zertifikate geltenden Anforderungen (bspw. GS-A_4357) angezogen werden.

3.12 kryptographische Vorgaben für die SAK des Konnektors

GS-A_5071 - kryptographische Vorgaben für eine Signaturprüfung in der SAK-Konnektor
Die SAK des Konnektors MUSS bei der Prüfung von qualifizierten elektronischen Signaturen mindestens folgende Verfahren wie im Algorithmenkatalog [ALGCAT] benannt, unterstützen:

- SHA-256, SHA-512/256, SHA-384, SHA-512 nach FIPS-180-4 (März 2012) [FIPS-180-4] (jeweils Abschnitt 6.2, 6.7, 6.5 und 6.4 ebenda),
- RSASSA-PSS nach PKCS#1 (PKCS#1 v2.1: RSA Cryptographic Standard, 14.06.2002) Abschnitt 8.1 und 9.1,
- RSASSA-PKCS1-v1_5 nach PKCS#1 (PKCS#1 v2.1: RSA Cryptographic Standard, 14.06.2002) Abschnitt 8.2 und 9.2,
- bei RSA muss ein Modulus zwischen 1976 bis 4096 Bit verwendbar sein,
- ECDSA basierend auf E(F_p) (vgl. Technische Richtlinie 03111, Version 2.0) auf der Kurve P256r1 [RFC-5639].

[<=]

3.13 Migration im PKI-Bereich

~~*Diese Vorgabe ist aus den Produkten TSP-CVC, TSP-X.509-nonQES, TSL-Dienst hier her verlagert worden (ehemals TIP1-A_2623).*~~

GS-A_5079 - Migration von Algorithmen und Schlüssellängen bei PKI-Betreibern
Der Anbieter einer Schlüsselverwaltung MUSS neue Vorgaben zu Algorithmen und/oder Schlüssellängen der gematik nach einer vorgegebenen Übergangsfrist umsetzen. Nach Ablauf der Übergangsfrist MÜSSEN ausschließlich diese geänderten Parameter bei der Erzeugung von Zertifikaten verwendet werden.**[<=]**

3.14 Spezielle Anwendungen von kryptographischen Signaturen

GS-A_5207 - Signaturverfahren beim initialen Pairing zwischen Konnektor und eHealth-Kartenterminal
Alle Produkttypen, die beim initialen Pairing zwischen Konnektor und eHealth-Kartenterminal die Signatur des Shared-Secret (ShS.AUT.KT vgl. [gemSpec_KT#2.5.2.1, 3.7.2.1]) erzeugen oder prüfen, MÜSSEN dafür RSASSA-PSS [PKCS#1] verwenden.
[<=]

Erläuterung: Beim initialen Pairing zwischen Konnektor und eHealth-Kartenterminal wird vom Konnektor ein 16 Byte langes Geheimnis erzeugt, das bei späteren Verbindungsaufbauten zwischen Konnektor und KT im Rahmen eines Challenge-Response-Verfahrens ([gemSpec_KT#3.7.2]) verwendet wird. Dieses Geheimnis wird von der gSMC-KT des KT beim initialen Pairing signiert. Die Signatur wird vom KT zum Konnektor transportiert und dort vom Konnektor geprüft.

GS-A_5208 - Signaturverfahren für externe Authentisierung
Der Konnektor MUSS an der Schnittstelle für die externe Authentisierung die Signaturverfahren RSASSA-PKCS1-v1_5 [PKCS#1] und RSASSA-PSS [PKCS#1] anbieten.**[**<=**]**

Erläuterung: Der Konnektor erlaubt (bei entsprechender Berechtigung) die direkte Nutzung der privaten Schlüssel MF/ DF.ESIGN/ PrK.HP.AUT.* auf einem HBA oder MF/ DF.ESIGN/ PrK.HCI.AUT.* auf einer SMC-B durch ein Primärsystem. Dies wird fast immer für eine klientenseitige TLS-Authentisierung gegenüber einem TLS-Server (außerhalb der TI) verwendet. Dafür werden über die Schnittstelle RSASSA-PKCS1-v1_5-Signaturen von den entsprechenden Karten erzeugt und über den Konnektor an ein Primärsystem übergeben. Für unbenannte Anwendungen müssen auch RSASSA-PSS-Signaturen erzeugbar sein. Diese Signaturen sind nicht als Dokumentensignaturen verwendbar, der Verwendungszweck ist in den zu den privaten Schlüsseln gehörigen Zertifikaten kodiert (ExtendedKeyUsage: keyPurposeId = id-kp-clientAuth).

GS-A_5340 - Signatur der TSL
Der TSL-Dienst MUSS für die Signatur der TSL das Signaturverfahren RSASSA-PSS [PKCS#1] verwenden mit dem XMLDSig-Identifier „http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1“ nach [RFC-6931, Abschnitt „2.3.10 RSASSA-PSS Without Parameters“].**[**<=**]**

3.15 ePA-spezifische Vorgaben

Die "vertrauenswürdige Ausführungsumgebung" (VAU) wird in [gemSpec_Dokumentenverwaltung] eingeführt. Jedes ePA-Frontend des Versicherten (FdV) muss mit jeder beliebigen VAU (egal von welchem Anbieter ePA-Aktensystem) kommunizieren können. Deshalb ist es für die Interoperabilität notwendig, dass Kommunikationsprotokoll zwischen beiden Kommunikationspartnern zu definieren und dessen Verwendung zu fordern.

A_15546 - ePA-Frontend des Versicherten: Kommunikation zwischen ePA-FdV und VAU
Das ePA-Frontend des Versicherten MUSS bei der Kommunikation mit der VAU das Kommunikationsprotokoll aus [gemSpec_Krypt#Abschnitt "Kommunikationsprotokoll zwischen VAU und ePA-Clients"] verwenden und dabei die Rolle Client einnehmen. Dabei MUSS es die CipherConfiguration "AES-128-GCM-BrainpoolP256r1-SHA-256" (vgl. Abschnitt 6) verwenden. Das ePA-Frontend des Versicherten MUSS nach spätestens 24 Stunden das Aushandeln eines neuen AES-Sitzungsschlüssels erzwingen. Es MUSS den abgelaufenen Sitzungsschlüssel bei sich sicher löschen.**[**<=**]**

Hinweis: ein ePA-Frontend des Versicherten ist nach A_15872 (bzw. A_15873) [gemSpec_Frontend_Vers] verpflichtet, das Zertifikat des Kommunikationspartners (VAU) zu prüfen (Kontext: Prüfung Authentizität des empfangene ECDH-Schlüssels). Nach A_15873 (vgl. auch A_15784) [gemSpec_Frontend_Vers] muss dabei die TSL der TI Prüfungsgrundlage sein [gemSpec_Frontend_Vers].

A_15549 - ePA-FM: Kommunikation zwischen ePA-FM und VAU
Das ePA-Fachmodul MUSS bei der Kommunikation mit der VAU das Kommunikationsprotokoll aus [gemSpec_Krypt#Abschnitt "Kommunikationsprotokoll zwischen VAU und ePA-Clients"] verwenden und dabei die Rolle Client einnehmen. Dabei MUSS es die CipherConfiguration "AES-128-GCM-BrainpoolP256r1-SHA-256" (vgl. Abschnitt 6) verwenden.
Das ePA-Fachmodul MUSS nach spätestens 24 Stunden das Aushandeln eines neuen AES-Sitzungsschlüssels erzwingen. Es MUSS den abgelaufenen Sitzungsschlüssel bei sich sicher löschen.**[**<=**]**

A_15561 - ePA-FM: AES-NI
Wenn der eingesetzte Konnektor AES-NI unterstützt und AES-NI dort aktiviert ist (vgl. [BSI-TR-03116-1#Abschnitt "4.7 Hardware-Unterstützung AES (AES-NI)"]), MUSS das ePA-Fachmodul bei der Umsetzung von A_15549 ("ePA-FM: Kommunikation zwischen ePA-FM und VAU") ebenfalls AES-NI verwenden.
[<=**]**

A_15547 - VAU: Kommunikation zwischen VAU und ePA-FdV bez. ePA-FM
Das ePA-Aktensystem MUSS sicherstellen, dass dessen VAU bei der Kommunikation mit dem ePA-Frontend des Versicherten oder dem ePA-FM das Kommunikationsprotokoll aus [gemSpec_Krypt#Abschnitt "Kommunikationsprotokoll zwischen VAU und ePA-Clients"] verwendet und dabei die Rolle Server einnimmt. Dabei MUSS es die CipherConfiguration"AES-128-GCM-BrainpoolP256r1-SHA-256" (vgl. Abschnitt 6) verwenden.
Die VAU MUSS nach spätestens 24 Stunden das Aushandeln eines neuen AES-Sitzungsschlüssels erzwingen. Die VAU MUSS den abgelaufenen Sitzungsschlüssel und das ephemere EC-Schlüsselpaar, das im ECDH Grundlage der Schlüsselableitung für diesen Schlüssel war, sicher löschen.
Die VAU MUSS zwei Zertifikate aus der Komponenten-PKI der TI besitzen (mit Rollenkennung-OID "oid_epa_vau"): eines mit einem RSA-EE-Schlüssel und eines mit einem ECC-EE-Schlüssel. Die VAU MUSS für die Erstellung der VAUServerHello-Nachricht mit beiden EE-Schlüssel signieren (Signatur der VAUHelloServerData). In der VAUHelloServer-Nachricht MUSS die VAU beiden EE-Zertifikate aufführen und die dazugehörigen OCSP-Responses.**[**<=**]**

Schlüsselexport:

A_15550 - ePA-Frontend des Versicherten: Export der für eine ePA notwendigen Schlüsseldaten
Ein ePA-Frontend des Versicherten MUSS beim Export der für eine ePA notwendigen Schlüsseldaten (Akten- und Kontextschlüssel) folgende Vorgaben durchsetzen:

1. Akten- und Kontext-Schlüssel MÜSSEN beim Export verschlüsselt werden. Es dürfen keine unverschlüsselten Schlüssel exportiert werden. Es MUSS dabei ein Verfahren der "Authenticated Encryption with Associated Data" verwendet werden, bei dem die Metadaten als assoziierte Daten integritätsgeschützt werden MÜSSEN.
2. Alle beim Export eingesetzten kryptographischen Verfahren (Verschlüsselungsverfahren, Schlüsselableitungsverfahren etc.) und deren verwendete Domainparameter (Schlüssellängen, Kurvenparameter etc.) MÜSSEN durch die [BSI-TR-03116-1] zulässig sein oder durch [BSI-TR-02102-1] empfohlen sein und DÜRFEN NICHT Legacy-Verfahren sein (bspw. SHA-1 innerhalb eines HMAC).
3. Falls Passwörter im Rahmen des Exports des Akten- und Kontextschlüssels verwendet werden, MÜSSEN die Vorgaben des BSI zur Regelung des Passwortgebrauchs (M2.11) erfüllt sein.

4. Es MUSS dem Nutzer die Möglichkeit angeboten werden, die zu exportierenden Schlüsseldaten in mehrere Teile mittels des Shamir-Secret-Sharing-Protokolls zu teilen [BSI-TR-02102-1, Abschnitt "8. Secret Sharing"]. Der Nutzer MUSS darauf hingewiesen werden, dass er die Sharings an unterschiedlichen Orten aufbewahren muss (vgl. auch Spiegelstrich 6).
5. Für die Entschlüsselung der exportierten Schlüsseldaten MÜSSEN mindestens zwei Sharing (vgl. (4)) oder mindestens zwei Faktoren (bspw. Besitz und Wissen) nötig sein.
6. Es MUSS dem Nutzer die Möglichkeit angeboten werden, exportierte Daten in einem Bildformat auf Papier auszudrucken in einer Form, die zukünftige Importe durch das ePA-Frontend des Versicherten wieder ermöglicht (bspw. 2D-Barcode).

[<=]

Erläuterung:
Die Verfügbarkeit eines 128-Bit-Schlüssels stellt in Sinne der Anforderung (A_15550) ein Beispiel für den Faktor Besitz dar, da es den meisten Menschen sehr schwer fällt mehr als 40 Bit Zufallsdaten über längere Zeit zu erinnern.
Die im Rahmen des Exports erzeugten Schlüssel und Zufallszahlen unterliegen aufgrund GS-A_4367 und GS-A_4368 ebendort festgelegten Mindestanforderungen.

A_15705 - Vorgaben Aktenschlüssel (RecordKey) und Kontextschlüssel (ContextKey)
Ein ePA-Frontend des Versicherten und ein ePA-FM MÜSSEN sicherstellen, dass

1. die von ihnen erzeugten Aktenschlüssel (RecordKey) und Kontextschlüssel (ContextKey) AES-Schlüssel [FIPS-197] mit 256 Bit Schlüssellänge sind,
2. diese Schlüssel von ihnen ausschließlich mittels AES/GCM analog [gemSpec_Krypt#GS-A_4373] bzw. [gemSpec_Krypt#GS-A_4389] verwendet werden und
3. sie die Arbeit mit Aktenschlüssel (RecordKey) und Kontextschlüssel (ContextKey), die nicht Spiegelstrich 1. erfüllen, ablehnen.

[<=]

A_15745 - Verschlüsselte Speicherung der verschlüsselten ePA-Daten
Ein ePA-Aktensystem MUSS sicherstellen, dass

1. es einen betreiberspezifischen Schlüssel (BS) gibt,
2. dieser Schlüssel ein AES-Schlüssel [FIPS-197] mit 256 Bit Schlüssellänge ist,
3. dieser Schlüssel in einem mindestens nach FIPS-140-2 Level 3 zertifizierten HSM liegt und nur dort verwendet wird,
4. dieser Schlüssel im Betrieb ausschließlich der VAU des entsprechenden ePA-Aktensystem zugänglich ist,
5. dieser Schlüssel nur zur Schlüsselableitung nach einem in [gemSpec_Krypt#Abschnitt 2.4] zulässigen Verfahren verwendet wird,
6. es eine Schlüsselableitung mit diesem betreiberspezifischen Schlüssel und einem aktenspezifischen Merkmal (bspw. der KVNR) gibt und daraus ein aktenspezifischer Schlüssel (ABS) abgeleitet wird,
7. dieser aktenspezifische Schlüssel ein AES-Schlüssel [FIPS-197] mit 256 Bit Schlüssellänge ist,
8. die verschlüsselten ePA-Daten einer Akte mit diesem aktenspezifischen Schlüssel verschlüsselt werden,

9. die verschlüsselten ePA-Daten außerhalb der VAU niemals im Klartext (also ohne mittels des ABS verschlüsselt zu sein) liegen,
10. dieser Schlüssel (ABS) ausschließlich mittels AES/GCM analog [gemSpec_Krypt#GS-A_4389] verwendet wird (der ABS wird durch Anfrage der VAU im HSM berechnet (Schlüsselableitung) und dann von dort an die VAU übermittelt, die AES/GCM-Operationen mit dem ABS finden in der VAU statt),
11. dieser Schlüssel (ABS) im Betrieb ausschließlich der VAU des entsprechenden ePA-Aktensystem zugänglich ist.

[<=]

Erläuterung: Das zu erreichende Ziel ist, dass, wenn ein Angreifer (mit hohem Angriffspotential, im Sinne von CC) selbst unter (1) der (hypothetischen) Annahme, die ePA an sich wäre überhaupt nicht verschlüsselt (es würde gar kein Aktenschlüssel existieren etc.) und (2), er alles im ePA-Aktensystem außer der VAU (inkl. HSM mit dem betreiberspezifischen Schlüssel) sicherheitstechnisch kompromittiert hätte, der Angreifer immer noch nicht auf die Klartextdaten zugreifen könnte.

Hintergrund ist, dass es unter dem Hybrid-Modell der ePA-Architektur aus Zugriffskontrolle und Verschlüsselung bei Entzug einer Berechtigung einem nun nicht mehr berechtigten Nutzer kryptographisch theoretisch immer noch möglich ist, die Daten der Akten, auf die er vormals berechtigten Zugriff hatte, zu entschlüsseln (er bricht bspw. in das Rechenzentrum des Anbieters ePA-Aktensystem ein und stiehlt dort alle Festplatten). Durch den in einem HSM beschützten betreiberspezifischen Schlüssel ist dieser beschriebene Angriff nun unterbunden.

A_15746 - Sicherstellung der Verfügbarkeit des betreiberspezifischen Schlüssels
Ein ePA-Aktensystem MUSS sicherstellen, dass für die Sicherstellung der Verfügbarkeit des betreiberspezifischen Schlüssels (vgl. A_15745) eine sicherheitstechnisch geeignete Sicherung des Schlüsselmaterials erzeugt und sicher verwahrt wird.

[<=]

A_15751 - TLS-Verbindung zwischen ePA-Aktensystem und ePA-FdV
Ein ePA-Aktensystem und ein ePA-Frontend des Versicherten MÜSSEN in Bezug auf die TLS-Verbindung zwischen ihnen

1. folgende Ciphersuiten unterstützen
   - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x30),
   - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2F)
   - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x2C),
   - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2B).
2. Sie KÖNNEN weitere Cipher-Suiten aus [TR-02102-2, Abschnitt 3.3.1 Tabelle 1] unterstützen.
3. Bei dem ephemeren Elliptic-Curve-Diffie-Hellman-Schlüsselaustausch und bei der Signaturprüfung mittels ECDSA MÜSSEN die Kurven P-256 oder P-384 [FIPS-186-4] unterstützt werden. Daneben SOLLEN die Kurven brainpoolP256r1, brainpoolP384r1 oder brainpoolP512r1 (vgl. [RFC-5639] und [RFC-7027]) unterstützt werden. Andere Kurven SOLLEN NICHT verwendet werden (Hinweis: die Intention des letzten Satzes ist insbesondere, dass die Ordnung des Basispunktes in E(F_p) nicht zu klein werden darf).

[<=]

A_15833 - TLS-Verbindungen ePA-FdV
Ein ePA-Frontend des Versicherten MUSS die TLS-Vorgaben in A_15751 bei allen seinen TLS-Verbindungen einhalten.
[<=]

A_16176 - Mindestvorgaben für ePA-Aktensystem-interne Schlüssel

Ein ePA-Aktensystem MUSS bei innerhalb des Aktensystems eingesetzten Schlüsselmaterial, das nicht aus der TI-PKI kommt (Signatur Authorisierungstoken etc.), folgende Vorgaben umsetzten:

1. Alle verwendeten nicht-TI-Schlüssel MÜSSEN ein Sicherheitsniveau von 120 Bit ermöglichen (vgl. [gemSpec_Krypt#5 "Migration 120-Bit Sicherheitsniveau"]).
2. Alle nicht-TI-RSA-Schlüssel MÜSSEN eine Mindestschlüssellänge von 3000 Bit besitzen.
3. Alle nicht-TI-ECC-Schlüssel MÜSSEN auf einem folgenden der Domainparametern (Kurven) basieren:
   a. P-256 oder P-384 [FIPS-186-4],
   b. brainpoolP256r1, brainpoolP384r1 oder brainpoolP512r1 [RFC-5639].

[<=]

Erläuterung: Ziel von A_15751 und A_16176 ist es, den Umstellungsbedarf im Rahmen der ECC-Migration der TI und ihrer Anwendungen in der Phase 2 zu minimieren.

4 Umsetzungsprobleme mit der TR-03116-1

Das u. a. durch die TR-03116-1 [BSI-TR-03116-1] angestrebte Sicherheitsniveau soll persönliche medizinische Daten effektiv schützen. Dazu lehnt sie sich an die sehr starken kryptographischen Vorgaben für die qualifizierte elektronische Signatur [SOG-IS-2018] an. Einige Formate (bspw. XMLDSig) oder Implementierungen (bspw. Standard-Java-Bibliotheken) können einige Vorgaben von Hause aus nicht erfüllen.

Dieses Kapitel weist auf Umsetzungsprobleme hin (ehemals Kapitel 3.3 aus dem Kryptographiekonzept des Basis-Rollouts).

4.1 XMLDSig und PKCS1-v2.1

Mit [XMLDSig] allein ist aktuell keine Nutzung von RSASSA-PSS [PKCS#1] möglich. Die Alternative für RSA-Signaturen RSASSA-PKCS1-v1_5 ist nach [BSI-TR-03116-1] nur noch bis Ende 2017 zulässig (insbesondere auch für digitale nicht-qualifizierte elektronische Signaturen).

Aus diesem Grund hat die gematik entschieden für die Signatur nach [XMLDSig] zusätzliche Identifier für RSASSA-PSS aus [RFC-6931] innerhalb der TI zu verwenden, welche auf der Lösung aus [XMLDSig-RSA-PSS] basieren. Der RFC-6931 [RFC-6931] ist die Aktualisierung von [RFC-4051]. Die in Abschnitt „2.3.9 RSASSA-PSS With Parameters“ und „2.3.10 RSASSA-PSS Without Parameters“ aufgeführten Identifier für RSASS-PSS-Signaturen müssen innerhalb von XMLDSig für solche Signaturen verwendet werden.

GS-A_5091 - Verwendung von RSASSA-PSS bei XMLDSig-Signaturen
Produkttypen, die RSASSA-PSS-Signaturen [PKCS#1] innerhalb von XMLDSig erstellen oder prüfen, MÜSSEN die Identifier aus [RFC-6931] Abschnitt „2.3.9 RSASSA-PSS With Parameters“ und „2.3.10 RSASSA-PSS Without Parameters“ für die Kodierung dieser Signaturen verwenden.
[<=]

Ein Beispiel aus [RFC-6931] Abschnitt „2.3.10 RSASSA-PSS Without Parameters“:

```
<SignatureMethod
     Algorithm=
     "http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1"
   />
```

Vgl. [gemSpec_COS, (N003.000)]: Die Hashfunktion, auf der die Mask-generation-function basiert, ist SHA-256 [FIPS-180-4]. Die Länge des salt ist gleich der Ausgabelänge eben jener Hashfunktion (= 256 Bit).

4.2 XMLEnc: Die Nutzung von RSAES-OAEP und AES-GCM

Bei der Verschlüsselung mittels XMLEnc [XMLEnc] gibt es zwei Probleme in Bezug auf fehlende Identifier für kryptographische Verfahren, die in Abstimmung mit dem BSI für den Einsatz in der TI notwendig sind.

- Für die symmetrische Verschlüsselung mittels AES-GCM ([FIPS-197], [NIST-SP-800-38D]) gibt es keine Algorithmen-Identifier innerhalb von [XMLEnc]. Solche gibt es in [XMLEnc-1.1, Abschnitt 5.2.4].
- Bei der Verschlüsselung mittels [PKCS#1] gibt es zwei Varianten: RSAES-OAEP und RSAES-PKCS1-v1_5. Beide Varianten werden von den Smartcards der TI unterstützt und für die zweite Variante stehen innerhalb von [XMLEnc] ausreichend Identifier zur Verfügung. Diese Variante ist nach [BSI-TR-03116-1] nur bis Ende 2017 zulässig. Bei der Variante RSAES-OAEP fehlt in [XMLEnc] ein Identifier für RSAES-OAEP mit der MGF basierend auf SHA-256 (vgl. auch Kapitel 5.10 „MGF Mask Generation Function“ in [gemSpec_COS]). Einen solchen Identifier(„http://www.w3.org/2009/xmlenc11#mgf1sha256“) gibt es in XMLEnc Version 1.1 [XMLEnc-1.1, Abschnitt 5.5.2].

Aus diesem Grund hat die gematik entschieden für die XML-Verschlüsselung die Vorgaben aus [XMLEnc-1.1] zu verwenden.

4.3 XML Signature Wrapping und XML Encryption Wrapping

Komplexität ist der natürliche Feind von Sicherheit. Die unten dem Sammelbegriff XML betitelten Formate und Protokolle sind sehr flexibel und leistungsfähig, aber auch sehr komplex. Noch dazu sind Sicherheitsmechanismen in diesem Bereich zum Teil nachträglich beigefügt worden und sind damit oft weniger leistungsfähig als im CMS-Bereich. XML-Daten effektiv zu schützen ist aktives Forschungsthema [XMLEnc-CM], [XSpRES]. Öfter als in anderen Bereichen werden neue Schwachstellen bekannt [BreakingXMLEnc], [XSW-Attack].

Aus diesem Grunde wird bei einer Sicherheitsevaluierung gesondert auf derartige Angriffe geachtet. Die gematik beobachtet neue Entwicklungen im Bereich der XML-Sicherheit und leitet falls notwendig Maßnahmen ein.

4.4 Güte von Zufallszahlen

Nach dem Kerckhoffs'schen Prinzip von 1883 [Ker-1883] darf die Sicherungsleistung von kryptographischen Verfahren alleinig auf der Geheimhaltung der geheimen oder privaten Schlüssel beruhen. Geheimhaltung inkludiert insbesondere, dass sie nicht erraten werden können. Wenn bei einer Schlüsselerzeugung zu wenig Entropie vorhanden ist, kann die Geheimhaltung nicht gewährleistet werden. Die kryptographischen Verfahren, welche mit diesen Schlüsseln dann arbeiten, können die von ihnen verlangten Sicherheitsleistungen nicht mehr erbringen. Aus diesem Grunde verlangt [BSI-TR-03116-1] eine Mindestgüte der Zufallszahlerzeugung u. a. bei einer Schlüsselerzeugung. Die Basis für die Beurteilung der Güte stellt [AIS-20] und [AIS-31] dar.

Aktuell sind nicht alle Produkte in der TI bez. dieser Mindestgüte bewertet worden. Davon sind Smartcards nicht betroffen, da diese eine Sicherheitsevaluierung/-zertifizierung durchlaufen haben, bei der die Güte der Zufallszahlenerzeugung positiv beurteilt wurde. Probleme bereiten insbesondere HSMs.

Neben einer möglichen Common-Criteria-Zertifizierung dieser Produkte, bei der analog zu den Smartcards die Güte geprüfte wird, gibt es weitere mögliche Lösungen:

1. gesonderte Prüfung der Güte nach [AIS-20] und [AIS-31] ohne komplette Common-Criteria-Zertifizierung,

2. Herstellererklärung über die Güte (wie sie bspw. aktuell bei der Kartenproduktion üblich ist).

5 Migration 120-Bit-Sicherheitsniveau

Das „Sicherheitsniveau eines kryptographischen Verfahrens“ ist definiert als der Logarithmus zur Basis 2 der Anzahl der „Rechenschritte“ die notwendig sind um ein kryptographisches Verfahren mit hoher Wahrscheinlichkeit zu brechen. Was als „Rechenschritt“ definiert ist, ist vom Verfahren abhängig. Das Sicherheitsniveau wird in Bit angegeben. Beispielsweise nimmt man aktuell an, dass für das Brechen einer AES-Chiffre mit 128 Bit Schlüssellänge rund $2^{126,4}$ Rechenschritte, die der Durchführung einer AES-Verschlüsselung (eines 128-Bit Eingabeblocks) entsprechen, im Mittel notwendig sind. Somit erreicht eine AES-128-Bit-Verschlüsselung maximal ein Sicherheitsniveau von ca. 126,4 Bit. Eine RSA-2048-Bit-Verschlüsselung erreicht ein Sicherheitsniveau von ca. 100 Bit.

Für den qualifizierten Vertrauensraum ist ab Ende 2024 [SOG-IS-2018] und für die TI ab Ende 2023 ein Sicherheitsniveau von mindestens 120 Bit für alle kryptographischen Verfahren vorgeschrieben [BSI-TR-03116-1]. Daher ist bis dahin eine Migration aller Komponenten und Dienste notwendig, die kryptographische Verfahren mit Schlüssellängen bez. Domainparametern verwenden die nur ein Sicherheitsniveau von unter 120 Bit erreichen können.

Aufgrund der höheren Performanz, insbesondere in Chipkarten und Embedded-Geräten, wird nicht auf RSA-3072-Bit sondern auf ECDSA mit 256-Bit-Schlüsseln migriert.

Die Migration erfolgt schrittweise und Komponenten und Dienste werden zusätzlich mit Schlüsselmaterial und Zertifikaten auf Basis von ECDSA auf der Kurve brainpoolP256r1 ausgestattet werden. Es gibt bis maximal Ende 2022 (vgl. Abschnitt 2.1.1.2) bzw. Ende 2023 (vgl. Abschnitt 2.1.1.1) ein Parallelbetrieb in der TI.

Zunächst werden die X.509-Root der TI (Produkttyp „gematik Root-CA“), die TSPs der TI und die Objektsysteme der Chipkarten verändert um einen späteren Parallelbetrieb zu ermöglichen. Erst in einer späteren Migrationsphase werden die neu erzeugten ECDSA-basierten Identitäten in Arbeitsabläufen der TI genutzt.

5.1 PKI-Begriff Schlüsselgeneration

In [gemKPT_PKI_TIP#3.2] wird der Begriff der Schlüsselgeneration eingeführt. Eine CA signiert Zertifikate im abstrahierten Sinne mit „ihrem Signaturschlüssel“. Dieser Schlüssel wird regelmäßig neu erzeugt und solange Verfahren und Schlüssellänge bzw. Domainparameter gleichbleiben, handelt es sich um eine neue Schlüsselversion. Kryptographisch betrachtet wurde der neue Signaturschlüssel zufällig (vgl. GS-A_4368) erzeugt, ist also kryptographisch unabhängig vom alten Signaturschlüssel, und die CA arbeitet mit mehreren kryptographischen Schlüsseln.

Beispiel: im Fall der X.509-Root der TI (vgl. Abschnitt 5.2) wird ihr Signaturschlüssel im Regelfall alle zwei Jahre neu erzeugt (vgl. GEM.RCA1 und GEM.RCA2, https://download.tsl.ti-dienste.de/). Der Signaturschlüssel liegt hier in zwei Versionen vor. Beide Schlüssel kommen aus der Schlüsselgeneration „RSA“.

Für die Migration muss ein Signaturschlüssel in der X.509-Root der TI erzeugt werden, der aus der Schlüsselgeneration „ECDSA“ stammt. Für ihn gelten die Vorgaben aus [gemSpec_Krypt#GS-A_4357, Schlüsselgeneration „ECDSA“].

5.2 X.509-Root der TI

Die X.509-Root der TI (Produkttyp: gematik Root-CA) ermöglicht es über eine klassische PKI-Baumstruktur die meisten Zertifikate der TI zu prüfen. Für zukünftige Anwendungen, die nur mit erhöhten Kosten das leistungsstarke, aber auch deutlich komplexere TSL-Modell auswerten können, ist sie eine Infrastrukturleistung der TI, so wie auch die CVC-Root.

Die X.509-Root muss für die Migration ECDSA-basierte Zertifikate für TSPs ausstellen können. Aufgrund von [gemSpec_PKI#GS-A_5511] muss die X.509-Root der TI neben dem Signaturschlüssel für die Schlüsselgeneration „RSA“ auch einen Signaturschlüssel für die Schlüsselgeneration „ECDSA“ gemäß GS-A_4357 (brainpoolP256r1) erzeugen, und diesen verwenden können.

Als Hilfestellung wird im Folgenden ein X.509-Root-TI-Zertifikat betrachtet. Gemäß GS-A_4357 muss der öffentliche ECDSA-Schlüssel der Schlüsselgeneration „ECDSA“ auf der Kurve brainpoolP256r1 liegen. Sei

```
d=SHA-256(„gemSpec_Krypt-Beispiel X.509-Root-TI ECDSA-Schluessel“)
=0x62e50dca4da29b0b10ead635a20b51fb1ec281d11f90cde8b5a9d92371ae8052
```

Dieses d wird als Ganzzahl (Little-Endian) interpretiert und dies sei der für das Beispiel maßgebliche private Schlüssel. Damit ergibt sich folgender öffentlicher Punkt auf der Kurve brainpoolP256r1:

```
(0x377434509adcbb827f74acd7adf0ce72aa28ddc53be3f15ea8023a9b0722c09d,
0x5364a99686c02092bbf9efde9878847b90f09d90b7ac4193553820258a58dfd5)
```

Folgend ist die ASN.1-DER-Kodierung des Schlüssels, so wie sie sich später auch im Zertifikat befindet, aufgeführt:

```
MFowFAYHKoZIzj0CAQYJKyQDAwIIAQEHA0IABDd0NFCa3LuCf3Ss163wznKqKN3FO+PxXqgCOpsH
IsCdU2SplobAIJK7+e/emHiEe5DwnZC3rEGTVTggJYpY39U=
```

```
  0  90: SEQUENCE {
  2  20:   SEQUENCE {
  4   7:     OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
 13   9:     OBJECT IDENTIFIER brainpoolP256r1 (1 3 36 3 3 2 8 1 1 7)
      :     }
24  66:   BIT STRING
      :     04 37 74 34 50 9A DC BB 82 7F 74 AC D7 AD F0 CE
      :     72 AA 28 DD C5 3B E3 F1 5E A8 02 3A 9B 07 22 C0
      :     9D 53 64 A9 96 86 C0 20 92 BB F9 EF DE 98 78 84
      :     7B 90 F0 9D 90 B7 AC 41 93 55 38 20 25 8A 58 DF
      :     D5
      :   }
```

Das selbstsignierte Beispiel-Root-Zertifikat im PEM-Format:

```
-----BEGIN CERTIFICATE-----
MIICajCCAg+gAwIBAgIBATAKBggqhkjOPQQDAjBtMQswCQYDVQQGEwJERTEVMBMG
A1UECgwMZ2VtYXRpayBHbWJIMTQwMgYDVQQLDCtaZW50cmFsZSBSb290LUNBIGRl
ciBUZWxlbWF0aWtpbmZyYXN0cnVrdHVyMREwDwYDVQQDDAhHRU0uUkNBMzAeFw0x
NjEyMDkwODQxNTZaFw0yNjEyMDcwODQxNTZaMG0xCzAJBgNVBAYTAkRFMRUwEwYD
VQQKDAxnZW1hdGlrIEdtYkgxNDAyBgNVBAsMK1plbnRyYWxlIFJvb3QtQ0EgZGVy
IFRlbGVtYXRpa2luZnJhc3RydWt0dXIxETAPBgNVBAMMCEdFTS5SQ0EzMFowFAYH
KoZIzj0CAQYJKyQDAwIIAQEHA0IABDd0NFCa3LuCf3Ss163wznKqKN3FO+PxXqgC
OpsHIsCdU2SplobAIJK7+e/emHiEe5DwnZC3rEGTVTggJYpY39WjgZ4wgZswHQYD
VR0OBBYEFBERSneTkJZDKt3uLzjddI870TMmMEIGCCsGAQUFBwEBBDYwNDAyBggr
BgEFBQcwAYYmaHR0cDovL29jc3Aucm9vdC1jYS50aS1kaWVuc3RlLmRlL29jc3Aw
DwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwFQYDVR0gBA4wDDAKBggq
```

```
ghQATASBIzAKBggqhkjOPQQDAgNJADBGAiEApQ6qGHTx97IsdzgoWH9/W32yt4rk
udUis0xxGZ48YOUCIQCTQ4puol5YYIAZYk74mfid3JBOvMBV/XgPV2WpS/99yg==
-----END CERTIFICATE-----
```

Relativ am Anfang des Zertifikats befindet sich die OID gemäß GS-A_4357

```
16  10:   SEQUENCE {
18   8:     OBJECT IDENTIFIER ecdsaWithSHA256 (1 2 840 10045 4 3 2)
      :     }
```

Ab Offset 280 befindet sich der schon o. g. öffentlicher Schlüssel:

```
282  90:   SEQUENCE {
284  20:     SEQUENCE {
286   7:       OBJECT IDENTIFIER ecPublicKey (1 2 840 10045 2 1)
295   9:       OBJECT IDENTIFIER brainpoolP256r1 (1 3 36 3 3 2 8 1 1 7)
       :       }
306  66:     BIT STRING
       :       04 37 74 34 50 9A DC BB 82 7F 74 AC D7 AD F0 CE
       :       72 AA 28 DD C5 3B E3 F1 5E A8 02 3A 9B 07 22 C0
       :       9D 53 64 A9 96 86 C0 20 92 BB F9 EF DE 98 78 84
       :       7B 90 F0 9D 90 B7 AC 41 93 55 38 20 25 8A 58 DF
       :       D5
       :     }
```

Und am Ende des Zertifikats befindet sich die ECDSA-Signatur:

```
535  10:   SEQUENCE {
537   8:     OBJECT IDENTIFIER ecdsaWithSHA256 (1 2 840 10045 4 3 2)
       :     }
547  73:   BIT STRING, encapsulates {
550  70:     SEQUENCE {
552  33:       INTEGER
       :         00 A5 0E AA 18 74 F1 F7 B2 2C 77 38 28 58 7F 7F
       :         5B 7D B2 B7 8A E4 B9 D5 22 B3 4C 71 19 9E 3C 60
       :         E5
587  33:       INTEGER
       :         00 93 43 8A 6E A2 5E 58 60 80 19 62 4E F8 99 F8
       :         9D DC 90 4E BC C0 55 FD 78 0F 57 65 A9 4B FF 7D
       :         CA
       :       }
       :     }
       :   }
```

Wenn das oben augeführte Zertifikat sind in der Datei root.pem befindet, kann man bspw. mittels

```
openssl verify -check_ss_sig root.pem
```

die Signatur überprüfen und erhält als Ausgabe:

```
root.pem: C = DE, O = gematik GmbH, OU = Zentrale Root-CA der
Telematikinfrastruktur, CN = GEM.RCA3
 error 18 at 0 depth lookup:self signed certificate
OK
```

5.3 ECDSA-Schlüssel in X.509-Zertifikaten

GS-A_5518 - Prüfung Kurvenpunkte bei einer Zertifikatserstellung

Alle Produkttypen, die X.509-Zertifikate erstellen und dabei öffentliche Punkte auf einer elliptischen Kurve in diesen Zertifikaten bestätigen, MÜSSEN überprüfen, ob die zu bestätigenden Punkte auch auf der zugehörigen Kurve (im Regelfall brainpoolP256r1 [RFC-5639#3.4]) liegen. Falls nein, MUSS der Produkttyp eine Zertifikatsausstellung verweigern.
[<=**]**

6 Kommunikationsprotokoll zwischen VAU und ePA-Clients

6.1 Motivation

Die Architekturentscheidung, dass auf der Verbindungsstrecke zwischen einem ePA-Frontend eines Versicherten (Client) bzw. eines ePA-FM (Client) und einer VAU (Server) immer HTTP über ein Gateway als Kommunikationsprotokoll verwendet wird, hat Vorteile. Ein Nachteil ist, dass damit keine direkte TLS-Verbindung zwischen Client und Server möglich ist. Der TLS-Kanal terminiert am Gateway. Um die "letzten Meile" zwischen Gateway und VAU innerhalb des ePA-Aktensystems nicht ungeschützt zu lassen, wird das "VAU-Protokoll" eingeführt und verwendet. Es wird nicht der Weg gewählt HTTP über TLS über HTTP über TLS zu tunneln. Stattdessen wird das folgende Protokoll eingeführt als eine leichtgewichtige Sicherungsschicht zwischen einer VAU (Server) und einem ePA-Frontend eines Versicherten (Client) bzw. eines ePA-FM (Client). Der Server und der Client besitzen jeweils eine oder mehrere kryptographische Langzeitidentitäten. Diese sind Grundlage für einen beidseitig authentisierten ephemeren ECDH. Aus dem gemeinsamen ECDH-Geheimnis wird mittels einer HKDF ein AES-Schlüssel abgeleitet. Per AES-GCM werden nach dem Schlüsselaustausch alle ausgetauschten Nutzerdaten kryptographisch gesichert. Es gibt einen Nachrichtenzähler der vor Replay-Attacken schützt.

Die gematik stellt auf Anfrage eine Beispielimplementierung bereit.

6.2 Übersicht

Es gibt bei der Kommunikation, die das Protokoll steuert, zwei aktive und einen passiven Teilnehmer. Der Client initiiert die Kommunikation per HTTP-POST-Request. Der Server antwortet als HTTP-Server auf diesen Request mit einer HTTP-Response. Das Gateway leitet die Daten weiter und erscheint unsichtbar. Das Gateway erzwingt die Verwendung des HTTP-Protokolls.

Das Kommunikationsprotokoll hat als Hauptaufgabe die zwischen Client und Server (VAU) auszutauschenden Nutzerdaten vor dem Gateway in Bezug auf Vertraulichkeit, Authentizität, Integrität (inkl. Verhinderung von Replay-Attacken) zu schützen. Das Protokoll bietet (absichtlich) keinen Schutz der HTTP-Header-Informationen. Es ist bei der ePA-Architektur sichergestellt, dass eine Änderungen der HTTP-Header-Informationen keine negativen Auswirkungen auf die Vertraulichkeit, Integrität und Authentizität der Nutzerdaten hat. Hinweis zum besseren Verständnis: Zwischen Gateway und Client besteht immer eine TLS-Verbindung, so dass nur das Gateway bzw. nur ein Angreifer im Aktensystem selbst die Header-Informationen ändern könnte.

Ein Designziel ist es, den Verbindungsaufbau möglichst "menschenlesbar" zu gestalten und so die Implementierung und die Fehlersuche (u. a. auch im Betrieb) zu erleichtern. So sind die meisten Nachrichten einfache, menschenlesbare JSON-Objekte. Die im Protokoll definierten Fehlermeldungen (VAUServerError-Nachricht) haben den Hauptzweck, die manuelle Fehleranalyse zu erleichtern. Das VAU-Protokoll ist relativ einfach, so dass ein Client oder ein Server bei Auftreten eines Fehlers selten etwas

anderes tun kann, als die Protokollabarbeitung abzubrechen (und als Client einen neuen Protokolldurchlauf zu initiieren). Dies bedeutet: die Art der aufgetretenen Fehlernachricht ist für den Client i. d. R. irrelevant, weil die Handlungsoptionen sehr begrenzt sind. Erst bei der manuellen Fehleranalyse werden die differenzierten Fehlermeldungen hilfreich.

Viele Nachrichten und Datenfelder ähneln fachlich Bestandteilen aus dem TLS-Protokoll und werden absichtlich anders benannt, um Verwechselungen auf OSI-Layer 8 zu vermeiden.

Das Protokoll bietet die Möglichkeit, zukünftig verschiedene CipherConfiguration (im TLS-Protokoll entspricht dies den TLS-Cipher-Suiten) zu unterstützen; aktuell gibt es genau eine ("AES-128-GCM-BrainpoolP256r1-SHA-256").

Der Ablauf der Schlüsselaushandlung des VAU-Protokolls ist im folgenden Ablaufdiagramm dargestellt.

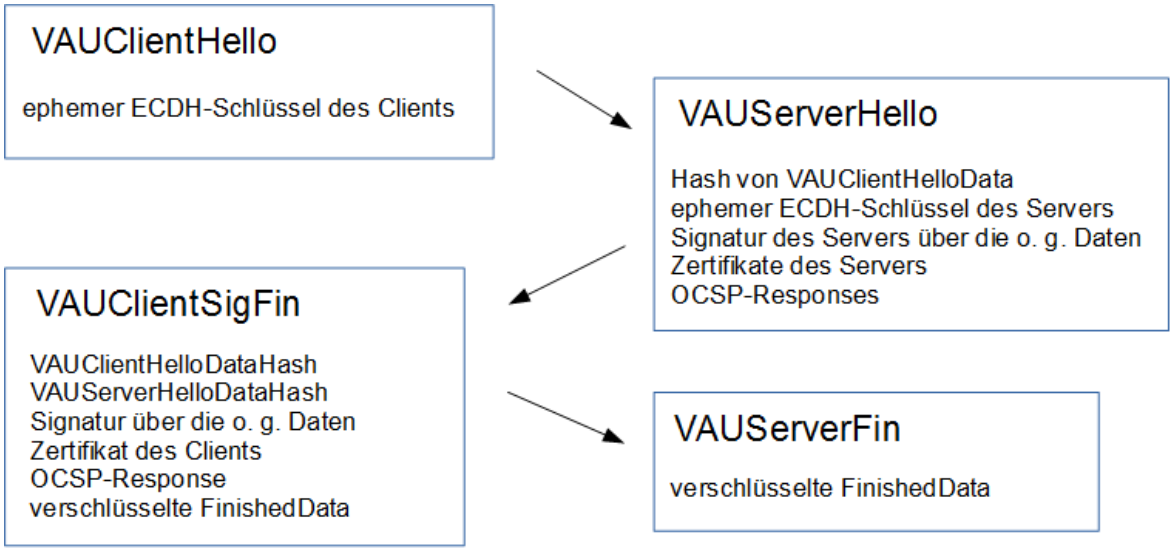

Abbildung 2: Übersicht über das VAU-Protokoll

Da die Werte des ephemeren öffentlichen ECDH-Schlüssels des jeweiligen anderen Kommunikationspartners in die eigene Signatur eingehen, können sowohl Client und Server bei der Signaturprüfung sicherstellen, dass die Schlüsselaushandlung "frisch" ist (Abschnitt "1.5 Freshness" [Boyd-Mathuria-2003]).

Aus dem gemeinsamen ECDH-Geheimnis wird ein AES-Schlüssel abgeleitet und damit die weitere Kommunikation mittels AES-GCM in Bezug auf Vertraulichkeit und Integrität geschützt. Der dabei explizit mitgelieferte Nachrichtenzähler schützt vor Replay-Attacken.

A_16884 - VAU-Protokoll: Nachrichtentypen und HTTP-Content-Type
Es gibt genau zwei Nachrichtenarten: (1) Nachrichten zur Schlüsselaushandlung (VAUClientHello, VAUServerHello, VAUClientSigFin und VAUServerFin) und Fehlermeldungsübermittlung (VAUServerError) und (2) Nachrichten, die kryptographisch geschützte Nutzerdaten transportieren.
Typ-(1)-Nachrichten MÜSSEN vom Client und vom Server jeweils per HTTP mit dem Content-Type 'application/json' übermittelt werden und Typ-(2)-Nachrichten mit dem Content-Type 'application/octet-stream'.**[<=]**

6.3 VAUClientHello-Nachricht

A_16883 - VAU-Protokoll: Aufbau VAUClientHello-Nachricht
Der Client MUSS die Kommunikation mittels einer VAUClientHello-Nachricht initiieren. Dafür erzeugt er zunächst eine VAUCientHelloData-Datenstruktur der Form

```
{
"DataType" : "VAUClientHelloData",
"CipherConfiguration" : [ "AES-128-GCM-BrainpoolP256r1-SHA-256" ],
"PubKey"  : "...PEM-kodierter-ephemerer-ECC-Schlüssel..."
}
```

Der Client MUSS im Rahmen der Schlüsselaushandlung ein ECDH-Schlüsselpaar basierend auf der Kurve BrainpoolP256r1 [RFC-5639] erzeugen. Er MUSS im "PubKey"-Feld den öffentliche Punkt des ephemeren ECDH-Schlüsselpaares im PEM-Format eintragen.
Beispiel:

```
{
"DataType" : "VAUClientHelloData",
"CipherConfiguration": [ "AES-128-GCM-BrainpoolP256r1-SHA-256" ],
"PubKey": "-----BEGIN PUBLIC KEY-----\nMFowFAY..."
}
```

Der Client MUSS diese Datenstruktur Base64-kodieren und vom Ergebnis einen SHA-256-Hashwert bilden, den er später mit dem im VAUServerHello aufgeführten Wert vergleichen werden muss. In das Datenfeld "Data" in der folgenden VAUClientHello-Nachricht MUSS er die kodierte VAUClientHelloData-Daten eintragen.

Die VAUClientHello-Nachricht hat folgenden Aufbau:

```
{
"MsgType"     : "VAUClientHello",
"Data"        : "...Base64-kodierte-VAUClientHelloData..."
}
```

[<=]

A_16897 - VAU-Protokoll: Versand der VAUClientHello-Nachricht
Der Client MUSS die VAUClientHello-Nachricht per HTTP mit dem Content-Type 'application/json' an den Server senden.**[<=]**

6.4 VAUServerHello-Nachricht

A_16898 - VAU-Protokoll: Erzeugung des Hashwert vom Data-Feld aus der VAUClientHello-Nachricht
Der Server MUSS beim Empfang der VAUClientHello-Nachricht einen SHA-256-Hashwert der Daten im Data-Feld (zunächst keine Base64-Dekodierung durchführen) erzeugen.**[<=]**

A_16895 - VAU-Protokoll: Zuordnung Signatur-IDs zu Signaturverfahren
Sowohl der Client als auch der Server MÜSSEN bei Signaturen mit dem Identifier

1. "ECDSAwithSHA256" ECDSA-Signaturen mittels der Hashfunktion SHA-256 erzeugen
   (Hinweis: da die Identitäten aus der PKI der TI kommen müssen, können die Signaturen immer nur auf Basis von BrainpoolP256r1 erzeugt bzw. geprüft werden) und
2. verbunden mit dem Identifier "RSASSA-PSSwithSHA256" RSASSA-PSS-Signaturen [RFC-8017] mittels der Hashfunktion SHA-256, mit der MGF1 (dort SHA-256 als Hashfunktion innerhalb der MFG) und einer Saltlänge von mindestens 256 Bit erzeugen.

[<=]

A_16901 - VAU-Protokoll: Aufbau der VAUServerHello-Nachricht
Der Server MUSS auf die VAUClientHello-Nachricht mit einer VAUServerHello-Nachricht, folgender Form, antworten

```
{
"MsgType"    : "VAUServerHello",
"Data"       : "...Base64-kodierte-Daten...",
"Signatures" : {
      "ECDSAwithSHA256" : "...hexadezimal-kodierte-ECDSA-Signatur...",
      "RSASSA-PSSwithSHA256" : "...hexadezimal-kodierte-RSASSA-PSS-
Signatur..."
},
"Certificates" : [
      "...PEM-kodiertes-Zertifikat-1...",
      "...PEM-kodiertes-Zertifikat-2..."
            ],
"OCSPResponses" : [
      "...Base64-kodierte-OCSP-Response-1...",
      "...Base64-kodierte-OCSP-Response-2..."
            ]
}
```

In den Daten von "Data" MUSS der Server in die Base64-kodierte VAUServerHelloData-Datenstruktur der folgenden Form eintragen.

```
{
"DataType" : "VAUServerHelloData",
"CipherConfiguration" : [ "AES-128-GCM-BrainpoolP256r1-SHA-256" ],
"VAUClientHelloDataHash" :  "...SHA-256-Hashwert-des-erhaltenen-Data-
Felds-in-VAUClientHello...",
"PubKey"  : "...PEM-kodierter-ephemerer-ECC-Schlüssel..."
}
```

Der Server MUSS im Feld "VAUClientHelloDataHash" den Base64-kodierten SHA-256-Hashwert der empfangenen VAUClientHelloData (ohne Base64-Dekodierung) eintragen (vgl. A_16898).

Der Server MUSS die in der Datenstruktur (VAUServerHello) angegebenen zwei Signaturen (vgl. A_16895) erzeugen (jeweils über den Base64-kodieren Wert im "Data"-Feld). Im "Certificates"-Feld MUSS er die für eine Signaturprüfung notwendigen EE-Zertifikate eintragen und im "OCSPResponses"-Feld die OCSP-Responses, die nicht älter als 24 Stunden sein dürfen, für diese EE-Zertifikate.**[<=]**

A_16902 - VAU-Protokoll: Versand der VAUServerHello-Nachricht
Der Server MUSS auf eine VAUClientHello-Nachricht mit einer VAUServerHello-Nachricht antworten. Der Server MUSS die VAUServerHello-Nachricht per HTTP mit dem Content-Type 'application/json' an den Client senden.**[<=]**

A_16903 - VAU-Protokoll: Client, Prüfung des VAUClientHelloDataHash-Werts (aus VAUServerHelloData)
Der Client MUSS beim Empfang der VAUServerHello-Nachricht den Hashwert "VAUClientHelloDataHash" (vgl. A_16901) mit dem vom ihm (Client) vor dem Versand der VAUClientHello-Nachricht (vgl. A 16883) errechneten Wert vergleichen. Sind die beiden Werte verschieden, so MUSS der Client den Protokollablauf abbrechen.**[<=]**

A_16941 - VAU-Protokoll: Client, Prüfung der Signatur der VAUServerHelloData
Der Client MUSS die Signatur der Daten von "Data" prüfen (bitgenau den Datenwert von "Data" nehmen, ohne Base64-Dekodierung). Der Client MUSS dafür den

Signaturschlüssel des Servers auf Authentizität und Integrität prüfen. (Hinweis: in einem Client wird die TSL der TI als Prüfgrundlage für die Prüfung von TI-Zertifikaten verwendet.) Falls die Signaturprüfung kein positives Ergebnis erbringt, so MUSS der Client den Protokollablauf abbrechen (vgl. A_16849).**[**<=**]**

6.5 Schlüsselableitung

A_16852 - VAU-Protokoll: ECDH durchführen
Der Client und auch der Server MÜSSEN jeweils für sich prüfen, ob der empfangene ephemere öffentliche elliptische Kurvenpunkt der Gegenseite auch auf der von ihnen verwendeten Kurve (BrainpoolP256r1) liegt.
Falls nein, MÜSSEN sie jeweils den Protokollablauf abbrechen. Falls der Server derjenige ist, der in diesem Fall abbricht, MUSS der zuvor an den Client eine VAUServerError-Nachricht mit der Fehlermeldung "invalid curve (ECDH)" senden.
Falls ja, MÜSSEN beide einen ECDH nach [NIST-800-56-A] durchführen. Das dabei erzeugte gemeinsame Geheimnis ist folgend Grundlage von zwei Schlüsselableitungen.**[**<=**]**

A_16943 - VAU-Protokoll: Schlüsselableitung (HKDF)
Für die Schlüsselableitung MÜSSEN Client und Server die HKDF nach [RFC-5869] auf Basis von SHA-256 verwenden. Die erste Schlüsselableitung hat den Ableitungsvektor 'KeyID' und erzeugt einen 256 Bit langen Schlüsselidentifier. Die zweite Schlüsselableitung mit dem Ableitungsvektor 'AES-128-GCM-Key' erzeugt den 128-Bit AES-Schlüssel für die Verwendung innerhalb von AES-128-GCM. **[**<=**]**

6.6 VAUClientSigFin-Nachricht

A_17070 - VAU-Protokoll: Aufbau der VAUClientSigFin-Nachricht
Der Client MUSS auf eine VAUServerHello-Nachricht mit einer wie folgt definierten VAUClientSigFin-Nachricht antworten.
Die VAUClientSigFin-Nachricht hat folgenden Aufbau:

```
{
"MsgType"    : "VAUClientSigFin",
"VAUClientHelloDataHash" : "...SHA-256-Hashwert-der-Base64-kodierten-
VAUClientHelloData...",
"VAUServerHelloDataHash" : "...SHA-256-Hashwert-der-erhaltenen-Base64-
kodierten-VAUServerHelloData...",
 "Signatures" : {
```

Entweder

```
     ECDSAwithSHA256"     : "...hexadezimal-kodierte-ECDSA-Signatur...",
```

oder

```
     RSASSA-PSSwithSHA256": "...hexadezimal-kodierte-RSASSA-PSS-
Signatur..."

 },
"Certificates": [ "...PEM-kodiertes-Zertifikat..." ],
"OCSPResponses" : [ "...Base64-kodierte-OCSP-Response-für-das-
Zertifikat-in-"Certificates"... ],
"FinishedData" : "...Base64-kodierte-verschlüsselte-Finished-Daten ..."
}
```

Im "VAUClientHelloDataHash"-Feld MUSS der Client den Base64-kodieren Hashwert seiner Base64-kodierten VAUClientHelloData eintragen.
Im "VAUClientServerDataHash"-Feld MUSS der Client den Base64-kodieren Hashwert der empfangenen Base64-kodierten VAUServerHelloData eintragen.
Die folgende Signatur MUSS der Client über die beiden konkatenierten Base64-kodierten Zeichenketten (Inhalt vom "VAUClientHelloDataHash"-Feld || Inhalt vom "VAUClientServerDataHash"-Feld) bilden.
Im "Signatures"-Feld MUSS entweder eine "ECDSAwithSHA256"- oder eine "RSASSA-PSSwithSHA256"-Signatur enthalten sein (vgl. A_16895 "VAU-Protokoll: Zuordnung Signatur-IDs zu Signaturverfahren").
Der Client MUSS im "Certificates"-Feld die für die Prüfung der Signatur notwendige X.509-EE-Zertifikat im PEM-Format eintragen.
Er SOLL für dieses Zertifikat im "OCSPResponses"-Feld die OCSP-Responses, die nicht älter als 24 Stunden sein dürfen, eintragen.
Der Client MUSS für die Berechnung des "FinishedData"-Feldes zunächst folgende Zeichenkette bilden

```
"VAUClientSigFin" ||
unkodierter Hashwert aus "VAUClientHelloDataHash" ||
unkodierter Hashwert aus  "VAUServerHelloDataHash"
```

Diese Zeichenkette MUSS 15+32+32=79 Bytes lang sein. Der Client MUSS diese Zeichenkette mittels AES-GCM (vgl. A_16943) verschlüsseln und dabei folgende Zeichenkette bilden

```
256-Bit KeyID || 96-Bit Nonce (IV) mit Ciphertext und 128 Bit Authentication-Tag
```

Diese Zeichenkette MUSS er Base64-kodieren und das Ergebnis als Wert des "FinishedData"-Feld eintragen.
[<=]

Hinweis: Obwohl innerhalb einer der VAUServerHelloData der Wert VAUClientHelloDataHash enthalten ist und damit auch in die Berechnung von VAUServerHelloDataHash mit einfließt, wird der Wert VAUClientHelloDataHash explizit in VAUClientSigFin aufgeführt. Ziel ist es möglichst direkt Transparenz über die bestätigten Daten zu schaffen.

A_17071 - VAU-Protokoll: Versand der VAUClientSigFin-Nachricht
Der Client MUSS auf eine VAUServerHello-Nachricht mit einer VAUClientSigFin-Nachricht antworten. Der Client MUSS die VAUClientSigFin-Nachricht per HTTP mit dem Content-Type 'application/json' an den Server senden.**[<=]**

6.7 VAUServerFin-Nachricht

A_17072 - VAU-Protokoll: Empfang der VAUClientSigFin-Nachricht
Der Server MUSS beim Empfang der VAUClientSigFin-Nachricht prüfen,

1. ob die darin enthaltene Signatur gültig ist, und
2. ob der Wert im "FinishedData"-Feld der nach A_17070 zu erwartenden Wert entspricht.

Falls die Signaturprüfung ein nicht-positives Prüferergebnis liefert, so MUSS der Server mit einer VAUServerError-Nachricht mit der Fehlermeldung "Signature from VAUClientSigFin invalid" antworten und die weitere Protokolldurchführung abbrechen.
Falls die Prüfung des "FinishedData"-Feld ein nicht-positives Prüferergebnis liefert, so MUSS der Server mit einer VAUServerError-Nachricht mit der Fehlermeldung "VAUClientSigFin invalid" antworten und die weitere Protokolldurchführung abbrechen.**[<=]**

A_16899 - VAU-Protokoll: Aufbau der VAUServerFin-Nachricht

Der Server MUSS eine wie folgt aufgebaute VAUServerFin-Nachricht erzeugen.

```
{
"MsgType"    : "VAUServerFin",
"FinishedData" : "...Base64-kodierte-verschlüsselte-Finished-Daten..."
}
```

Der Server MUSS für die Berechnung des "FinishedData"-Feldes zunächst folgende Zeichenkette bilden

```
"VAUServerFin" ||
unkodierter Hashwert aus "VAUClientHelloDataHash" ||
unkodierter Hashwert aus "VAUServerHelloDataHash"
```

Diese Zeichenkette MUSS 12+32+32=76 Bytes lang sein. Der Server MUSS diese Zeichenkette mittels AES-GCM (vgl. A_16943) verschlüsseln und dabei folgende Zeichenkette bilden

```
256-Bit KeyID || 96-Bit Nonce (IV) mit Ciphertext und 128 Bit Authentication-Tag
```

Diese Zeichenkette MUSS er Base64-kodieren und das Ergebnis als Wert des "FinishedData"-Feld eintragen.**[**<=**]**

A_17073 - VAU-Protokoll: Versand der VAUServerFin-Nachricht
Der Server MUSS nach dem Erhalt einer VAUClientSigFin-Nachricht mit einer VAUServerFin-Nachricht antworten. Der Server MUSS die VAUServerFin-Nachricht per HTTP mit dem Content-Type 'application/json' an den Client senden.**[**<=**]**

A_17084 - VAU-Protokoll: Empfang der VAUServerFin-Nachricht
Der Client MUSS beim Empfang der VAUServerFin-Nachricht prüfen, ob der Wert im "FinishedData"-Feld der nach A_16899 zu erwartenden Wert entspricht. Falls nein, so MUSS der Client den weiteren Protokollablauf abbrechen (vgl. A_16849). **[**<=**]**

6.8 Nutzerdatentransport

A_16945 - VAU-Protokoll: Client, verschlüsselte Kommunikation
Wie bei der Schlüsselaushandlung MUSS der Client mittels HTTP-POST-Request die nun verschlüsselte Kommunikation initiieren. Sei "Plaintext" die zu übermittelnde Nachricht. Der Client MUSS einen unsigned 64-Bit-Zähler führen, die er bei jeder abgeschickten Nachricht um zwei erhöhen MUSS. Er bildet die Datenstruktur "P1" mit

```
P1=Zähler-im-Big-Endian-Format || Plaintext
```

Der Zähler MUSS initial mit 1 starten. Der Client MUSS P1 mittels AES-GCM unter Verwendung einer zufällig erzeugten 96-Bit-Nonce verschlüsseln. Der Client berechnet so den "Ciphertext".
Die vom Client nun an den Server zu übermittelnde Datenstruktur MUSS folgende Form besitzen.

```
256-Bit KeyID || 96-Bit Nonce (IV) mit Ciphertext und 128 Bit Authentication-Tag
```

Diese Nachricht MUSS der Client per HTTP-POST-Request mit Content-Type 'application/octet-stream' ohne weitere Kodierungen versenden.**[**<=**]**

A_16952 - VAU-Protokoll: Server, verschlüsselte Kommunikation
Der Server erkennt aus der KeyID, welchen AES-Schlüssel er verwenden muss. Falls ihm die KeyID unbekannt ist, so MUSS er mit einer VAUServerError-Nachricht mit der Fehlermeldung "KeyID XXX not found" antworten, wobei er XXX durch die empfangene KeyID in Hexadezimalform ersetzen MUSS.
Falls bei der Entschlüsselung ein Fehler auftritt (bspw. Authentication-Tag passt nicht zur Nachricht), MUSS der Server mit einer VAUServerError-Nachricht mit der Fehlermeldung

"AES-GCM decryption error." antworten.

Falls die Entschlüsselung erfolgreich war, MUSS der Server die ersten 64-Bit (8 Byte) als Zähler (unsigned, im Big-Endian-Format) interpretieren und diese vom folgenden Plaintext entfernen. Der Zählerwert MUSS größer als der letzte auf diese Art empfangene Zählerwert (für die aktuell KeyID) plus 1 sein. (Zähler + 1 war der Zählerwert der Server-Response.)
Anderenfalls MUSS er

1. die KeyID in seiner Datenbasis verwerfen und alle damit verbundenen Schlüssel sicher löschen,
2. die restlichen Plaintext-Daten verwerfen, und
3. mit einer VAUServerError-Nachricht antworten mit der Fehlermeldung "Counter too small."

Anderenfalls (also Zählerwert ist OK) MUSS der Server seine Datenbasis in Bezug auf den mit der KeyID verbundenen Zähler aktualisieren und verarbeitet die Plaintext-Daten weiter.

Der Server erzeugt zunächst die Datenstruktur "P2" mit
`P2 = Zählerwert aus dem Request + 1 (unsigned 64-Bit, big-endian-format) || Klartext-Antwort des Servers`

Falls es zu einen Zählerüberlauf kommt, so MUSS der Server eine VAUServerError-Nachricht senden mit der Fehlermeldung "counter overflow", die KeyID in dessen Datenbasis verwerfen, die damit verbundenen Schlüssel sicher löschen und die Applikationsdaten ("Klartext-Antwort des Servers") verwerfen.

Der Server MUSS P2 mit AES-128-GCM verschlüsseln. Dafür MUSS der Server zufällig eine 96-Bit-Nonce erzeugen und bei der AES-GCM-Verschlüsselung verwenden. Die zu übermittelnde Datenstruktur MUSS folgende Form besitzen
`256-Bit KeyID || 96-Bit Nonce (IV) mit Ciphertext und 128 Bit Authentication-Tag`

Diese Datenstruktur MUSS der Server per HTTP-Response mit Content-Type 'application/octet-stream' ohne weitere Kodierungen versenden.**[**<=**]**

A_16957 - VAU-Protokoll: Client, verschlüsselte Kommunikation
Beim Empfang der Antwort (vgl. A_16952) MUSS der Client folgende Vorgaben durchsetzen.
Falls

1. er die KeyID nicht kennt,
2. die Entschlüsselung fehlschlägt (bspw. Authentication-Tag passt nicht zur Nachricht), oder
3. der 64-Bit Zählerwert ungleich 1 plus dem Zählerwert ist, den der Client für den Request verwendet hat,

so MUSS der Client die Nachricht verwerfen und die weitere Protokollausführung mittels des empfangenen KeyID abbrechen.

Anderen falls (alles ok, kein Abbruch) MUSS der Client den mit der KeyID verbundenen Zählerwert um eins erhöhen. D. h., Nachrichten vom Client an den Server haben immer einen ungeraden Zählerwert.
[<=**]**

A_16958 - VAU-Protokoll: Client, Neuinitiieren einer Schlüsselaushandlung
Der Client KANN jeder Zeit eine neue Schlüsselaushandlung (VAUClientHello etc.) initiieren.
[<=]

A_17069 - VAU-Protokoll: Client Zählerüberlauf
Der Client MUSS, falls so viele Nachrichten ausgetauscht werden, dass für den unsigned 64-Bit-Nachrichtenzähler ein arithmetischer Überlauf droht, eine neue Schlüsselaushandlung initiieren (VAUClientHello etc.).**[<=]**

6.9 VAUServerError-Nachricht

A_16851 - VAU-Protokoll: VAUServerError-Nachrichten
Der Server MUSS folgende Vorgaben umsetzen:
In verschiedenen im Protokoll beschriebenen Fehlerfällen sendet der Server eine VAUServerError-Nachricht an den Client.
Für die eigentliche Fehlerübermittlung MUSS folgende Datenstruktur erzeugt:

```
{
"DataType" : "VAUServerErrorData",
"Data"     : "...Fehlermeldung...",
"Time"     : "...aktuelle-Zeit-in-der-VAU..."
}
```

Die Zeit im "Time"-Feld MUSS im Format nach ISO-8601 kodiert werden (Beispiel: "2018-11-22-T10:00:00.123456").
Diese Datenstruktur MUSS der Server Base64-kodieren und in der folgenden Nachricht im Datenfeld "Data" einbetten.

```
{
"MsgType" : "VAUServerError",
"Data" : "...Base64-kodierte-VAUServerErrorData..",
"Signatures" : {
     "ECDSAwithSHA256"     : "...hexadezimal-kodierte-ECDSA-
Signatur...",
     "RSASSA-PSSwithSHA256": "...hexadezimal-kodierte-ECDSA-
Signatur..."
},
"Certificates": [
     "... PEM-kodiertes-Zertifikat-1 ...",
     "... PEM-kodiertes-Zertifikat-2 ..."
],
"OCSPResponses" : [
     "...Base64-kodierte-OCSPResponse-1...",
     "...Base64-kodierte-OCSPResponse-2..."
]
}
```

Der Server MUSS zwei Signaturen (vgl. "Signatures"-Feld und A_16895) erzeugen (jeweils über den Base64-kodierten Wert im "Data"-Feld) und bei der Erzeugung der VAUServerError-Nachricht im "Signatures"-Feld eintragen.
Im "Certificates"-Feld MUSS er die für eine Signaturprüfung notwendigen EE-Zertifikate eintragen und im "OCSPResponses"-Feld die OCSP-Responses, die nicht älter als 24 Stunden sein dürfen, für diese EE-Zertifikate.**[<=]**

A_16900 - VAU-Protokoll: Client, Behandlung von Fehlernachrichten

Erhält der Client eine VAUServerError-Nachricht (vgl. A_16851), MUSS er die Signatur prüfen. Falls die Prüfungen positiv ist, so MUSS er die Protolldurchführung abbrechen (vgl. A_16849).**[<=]**

6.10 Behandlung von parallelen Signaturen und zusätzlichen Datenfeldern

Verschiedene Typ-(1)-Nachrichten (VAUServerHello, VAUServerError) enthalten parallel RSASSA-PSS- und ECDSA-Signaturen. Der fachlicher Hintergrund dafür ist, dass die Mehrzahl der Konnektoren noch keine ECDSA-Signaturen im Feld prüfen können, da der dafür notwendige TSL-Dienst "ECC+RSA" sich noch im Aufbau befindet.

A_16850 - VAU-Protokoll: Behandlung von parallelen Signaturen
Ein Client oder ein Server MUSS, wenn

1. er ECDSA-Signaturen prüfen kann (Prüfungsgrundlage TSL-Dienst "ECC+RSA" ist vorhanden und für ihn nutzbar), und
2. in einer Typ-(1)-Nachricht RSA-basierte Signaturen und ECDSA-Signaturen parallel vorhanden sind,

genau nur die ECDSA-Signatur prüfen. Die RSA-basierte Signatur MUSS er dann ignorieren.**[<=]**

A_17074 - VAU-Protokoll: Ignorieren von zusätzlichen Datenfeldern in Protokoll-Nachrichten
Ein Client oder ein Server MUSS zusätzliche (i. S. v. ihm unbekannte) Datenfelder (Key-Value-Paare) in JSON-Objekten (Typ-(1)-Nachrichten und "Data"-Feldern darin) im Rahmen des VAU-Protokolls ignorieren.
[<=]

A_17081 - VAUProtokoll: zu verwendende Signaturschlüssel
Ein Client und ein Server MUSS für die Signatur im Rahmen des VAU-Protokolls (VAUServerHello- und VAUClientSigFin-Nachrichten) Schlüsselmaterial verwenden, dass dediziert für die Entity-Authentication vorgesehen ist (AUT-Schlüsselmaterial (einer eGK, einer SMC-B etc.) oder privates Schlüsselmaterial der VAU-Server-Identität (Rollenprofil "oid_epa_vau")).
[<=]

6.11 Abbrechen des Protokollablaufs

A_16849 - VAU-Protokoll: Aktionen bei Protokollabbruch
Wenn ein Client oder ein Server den Protokollablauf nach Protokollbeschreibung abbrechen muss, dann MUSS dieser die eventuell aktuell vorhandene KeyID aus seiner Datenbasis löschen und die damit verbundenen Schlüssel sicher löschen.
[<=]

7 Anhang – Verzeichnisse

7.1 Abkürzungen

Kürzel	Erläuterung
C2C	Card to Card
C2S	Card to Server
CA	Certificate Authority
CBC	Cipher Block Chaining
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
DRNG	Deterministic Random Number Generator
eGK	elektronische Gesundheitskarte
IV	Initialisierungsvektor
MAC	Message Authentication Code
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OSI	Open Systems Interconnection
SAK	Signaturanwendungskomponente
SM	Service-Monitoring
TI	Telematikinfrastruktur
TLS	Transport Layer Security
TSIG	Transaction Signature
URI	Uniform Resource Identifier

7.2 Glossar

Das Glossar wird als eigenständiges Dokument, vgl. [gemGlossar] zur Verfügung gestellt.

7.3 Abbildungsverzeichnis

7.4 Tabellenverzeichnis

7.5 Referenzierte Dokumente

7.5.1 Dokumente der gematik

Die nachfolgende Tabelle enthält die Bezeichnung der in dem vorliegenden Dokument referenzierten Dokumente der gematik zur Telematikinfrastruktur. Der mit der vorliegenden Version korrelierende Entwicklungsstand dieser Konzepte und Spezifikationen wird pro Release in einer Dokumentenlandkarte definiert, Version und Stand der referenzierten Dokumente sind daher in der nachfolgenden Tabelle nicht aufgeführt. Deren zu diesem Dokument passende jeweils gültige Versionsnummer sind in der aktuellsten, von der gematik veröffentlichten Dokumentenlandkarte enthalten, in der die vorliegende Version aufgeführt wird.

[Quelle]	Herausgeber: Titel
[gemGlossar]	gematik: Glossar der Telematikinfrastruktur
[gemSpec_COS]	gematik: Spezifikation des Card Operating System (COS)
[gemSpec_eGK_ObjSys]	gematik: Die Spezifikation der elektronischen Gesundheitskarte (eGK) – Objektsystem
[gemSpec_KT]	gematik: Spezifikation eHealth-Kartenterminal
[gemSpec_SST_FD_VSDM]	gematik: Schnittstellenspezifikation Fachdienste (UFS/VSDD/CMS)

7.5.2 Weitere Dokumente

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[AIS-20-1999]	W. Schindler: Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators. Version 1.0, 02.12.1999, ehemalige mathematisch technische Anlage zur AIS20, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Zertifierung/Interpretation/AIS20_Functionality_Classes_Evaluation_Methodology_DRNG.pdf?__blob=publicationFile
[AIS-20]	AIS 20: Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren, Version 3, 15.05.2013, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifierung/Interpretation/AIS_20_pdf.pdf?__blob=publicationFile
[AIS-31]	AIS 31: Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren, Version 3, 15.05.2013, http://www.bsi.bund.de/SharedDocs/Downloads/DE/BS/Zertifierung/Interpr

	etationen/ AIS_31_pdf.pdf?__blob=publicationFile
[ALGCAT]	Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen), Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, vom 30.12.2016 (auch online verfügbar: https://www.bundesanzeiger.de mit dem Suchbegriff „BAnz AT 30.12.2016 B5“)
[ANSI-X9.31]	National Institute of Standards and Technology, NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, January 31, 2005. http://csrc.nist.gov/groups/STM/cavp/documents/rng/931rngext.pdf
[ANSI-X9.62]	ANSI X9.62:2005 Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)
[ANSI-X9.63]	American National Standard for Financial Services X9.63–2001 Public Key Cryptography for the Financial Services Industry Key Agreement and Key Transport Using Elliptic Curve Cryptography
[Boyd-Mathuria-2003]	Protocols for Authentication and Key Establishment, Colin Boyd and Anish Mathuria, 2003
[BrainPool]	ECC Brainpool Standard Curves and Curve Generation v. 1.0 19.10.2005 http://www.teletrust.de/fileadmin/files/oid/oid_ECC-Brainpool-Standard-curves-V1.pdf
[Breaking-TLS]	Lucky Thirteen: Breaking the TLS and DTLS Record Protocols Nadhem J. AlFardan and Kenneth G. Paterson Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK, 6th February 2013
[BreakingXMLEnc]	How to Break XML Encryption, Tibor Jager, Juraj Somorovsky, 2011 http://www.nds.rub.de/media/nds/veroeffentlichungen/2011/10/22/HowToBreakXMLenc.pdf
[BSI-TR-02102-1]	BSI TR-02102-1 Technische Richtlinie „Kryptographische Verfahren: Empfehlungen und Schlüssellängen“ Version 2018-02, Stand 29.05.2018 https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html
[BSI-TR-02102-2]	BSI TR-02102-3 Technische Richtlinie „Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 2 – Verwendung von Transport Layer Security (TLS), Version 2018-01 https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html
[BSI-TR-	BSI TR-02102-3 Technische Richtlinie „Kryptographische Verfahren:

02102-3]	Empfehlungen und Schlüssellängen, Teil 3 – Verwendung von Internet Protocol Security (IPsec) und Internet Key Exchange (IKEv2)“ Version 2018-01 https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html
[BSI-TR-03116-1]	Technische Richtlinie BSI TR-03116-1 Kryptographische Vorgaben für Projekte der Bundesregierung, Version: 3.20, Fassung September 2018, 21.09.2018 https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03116/index_htm.html
[CM-2014]	20 Years of SSL/TLS Research, An Analysis of the Internet’s Security Foundation, Christopher Meyer, 9. February 2014 http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/MeyerChristopher/diss.pdf
[eIDAS]	Verordnung (EU) Nr. 910/2014 des europäischen Parlaments und des Rates vom 23. Juli 2014 über elektronische Identifizierung und Vertrauensdienste für elektronische Transaktionen im Binnenmarkt und zur Aufhebung der Richtlinie 1999/93/EG
[EN-14890-1]	DIN EN 14890-1:2008 Application Interface for smart cards used as Secure Signature Creation Devices - Part 1: Basic services
[ETSI-CAdES]	ETSI TS 101 733 V1.7.4 (2008-07), Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)
[ETSI-XAdES]	ETSI TS 101 903 V1.4.2 (2010-12), Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)
[FIPS-180-4]	Federal Information, Processing Standards Publication 180-4, Secure Hash Standard (SHS), March 2012 http://csrc.nist.gov/publications/fips/fips180-4/fips180-4.pdf
[FIPS-186-2+CN1]	FIPS 186-2 - National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2, January 27, 2000 – Appendix 3.1 unter der Beachtung des Change Notice 1, vom 5. Oktober 2001 http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2-change1.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
[IR-2014]	Bulletproof SSL and TLS: Understanding and deploying SSL/TLS and PKI to secure servers and web applications, Ivan Ristić, 2014 https://www.feistyduck.com/books/bulletproof-ssl-and-tls/
[ISO-11770]	ISO/IEC 11770: 1996, Information technology – Security techniques –

	Key management, Part 3: Mechanisms using asymmetric techniques
[Ker-1883]	Auguste Kerckhoffs, "La cryptographie militaire", Journal des sciences militaires, vol. IX, Seite 5–83, Jan. 1883, Seite 161–191, Feb. 1883. siehe auch http://www.petitcolas.net/fabien/kerckhoffs/
[KS-2011]	W. Killmann, W. Schindler, „A proposal for: Functionality classes for random number generators“, Version 2.0, September 2011 https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Zertifierung/Interpretation /AIS31_Functionality_classes_for_random_numder_generators.pdf?__blpb=publicationFile
[NIST SP-800-38A]	NIST Special Publication 800-38A, Recommendation for Block, Cipher Modes of Operation, Methods and Techniques, Morris Dworkin, December 2001 Edition, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
[NIST SP-800-38B]	NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Morris Dworkin, May 2005 Edition, http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
[NIST-SP-800-38D]	NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, Morris Dworkin, November, 2007
[NIST-SP-800-56-A]	NIST Special Publication 800-56A Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, ~~March, 2007~~ April 2018 https://csrc.nist.gov/publications/detail/sp/800-56a/rev-3/final
[NIST-SP-800-56-B]	NIST Special Publication 800-56B Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, August 2009
[NIST-SP-800-56C]	NIST Special Publication 800-56C Recommendation for Key Derivation through Extraction-then-Expansion, November 2011
[NIST-SP-800-108]	NIST Special Publication 800-108 Recommendation for Key Derivation Using Pseudorandom Functions, October 2009
[NK-PP]	Common Criteria Schutzprofil (Protection Profile) Schutzprofil 1: Anforderungen an den Netzkonnektor, BSI-CC-PP-0097
[Oorschot-Wiener-1996]	On Diffie-Hellman Key Agreement with Short Exponents, Paul C. van Oorschot, Michael J Weiner, Eurocrypt’ 96
[Padding-	Padding Oracle Attacks on CBC-mode Encryption with Secret and

Oracle-2005]	Random IVs Arnold K. L. Yau, Kenneth G. Paterson and Chris J. Mitchell, FSE 2005 http://www.isg.rhul.ac.uk/~kp/secretIV.pdf
[PAdES-3]	ETSI TS 102 778-3 V1.2.1, PDF Advanced Electronic Signature Profiles; Part 3: PAdES Enhanced – PAdES-BES and PAdES-EPES Profiles Technical Specification, 2010
[PDF/A-2]	ISO 19005-2:2011 – Document management – Electronic document file format for long-term preservation – Part 2: Use of ISO 32000-1 (PDF/A-2)
[PP-0082]	Common Criteria Protection Profile, Card Operating System Generation 2 (PP COS G2), BSI-CC-PP-0082-V2, Version 1.9, 18th November 2014
[RFC-2119]	RFC 2119 (März 1997): Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, http://tools.ietf.org/html/rfc2119
[RFC-2590]	RFC 2590 (June 1999): X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP https://tools.ietf.org/html/rfc2560 (Obsoleted by [RFC-6960])
[RFC-3279]	RFC 3279 (April 2002): Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile https://tools.ietf.org/html/rfc3279
[RFC-3447], [PKCS#1]	"Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", February 2003 https://tools.ietf.org/html/rfc3447
[RFC-3526]	RFC 3526 (Mai 2003: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) http://tools.ietf.org/html/rfc3526
[RFC-4051]	Additional XML Security Uniform Resource Identifiers (URIs), April 2005 https://tools.ietf.org/html/rfc4051
[RFC-4635]	RFC 4635 (August 2006): HMAC SHA TSIG Algorithm Identifiers http://tools.ietf.org/html/rfc4635
[RFC-5077]	Transport Layer Security (TLS) Session Resumption without Server-Side State, January 2008, https://tools.ietf.org/html/rfc5077
[RFC-5084]	RFC 5084: Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS), November 2007 https://tools.ietf.org/html/rfc5084
[RFC-5246]	The Transport Layer Security (TLS) Protocol Version 1.2, August 2008, https://tools.ietf.org/html/rfc5246

[RFC-5280]	RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Mai 2008 https://tools.ietf.org/html/rfc5280
[RFC-5480]	RFC 5480 (March 2009): Elliptic Curve Cryptography Subject Public Key Information, https://tools.ietf.org/html/rfc5480
[RFC-5639]	RFC 5639 (March 2010): Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, http://www.ietf.org/rfc/rfc5639.txt
[RFC-5652]	RFC 5652 (September 2009): Cryptographic Message Syntax (CMS), R. Housley, http://tools.ietf.org/html/rfc5652
[RFC-5702]	RFC 5702 (October 2009): Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC, http://tools.ietf.org/html/rfc5702
[RFC-5746]	Transport Layer Security (TLS) Renegotiation Indication Extension, February 2010, https://tools.ietf.org/html/rfc5746
[RFC-5869]	HMAC-based Extract-and-Expand Key Derivation Function (HKDF), May 2010, https://tools.ietf.org/html/rfc5869
[RFC-931]	RFC 6931: Additional XML Security Uniform Resource Identifiers (URIs), Donald Eastlake, April 2013, https://tools.ietf.org/html/rfc6931
[RFC-6960]	RFC 6960 (June 2013): X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, https://tools.ietf.org/html/rfc6960
[RFC-7027]	RFC 7027: (October 2013) Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS), https://tools.ietf.org/html/rfc7027
[RFC-7296]	RFC 7296 (October 2014): Internet Key Exchange Protocol Version 2 (IKEv2), https://tools.ietf.org/html/rfc7296
[RFC-7427]	RFC 7427 (January 2015): Signature Authentication in the Internet Key Exchange Version 2 (IKEv2), https://tools.ietf.org/html/rfc7427
[SDH-2016]	Measuring the Security Harm of TLS Crypto Shortcuts, Drew Springall, Zakir Durumeic, J. Alex Halderman, November 2016, https://jhalderm.com/pub/papers/forward-secrecy-imc16.pdf
[SOG-IS-2018]	SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms, Version 1.1, June 2018 https://www.sogis.org/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.1.pdf
[TLS-Attacks]	Lessons Learned From Previous SSL/TLS Attacks - A Brief Chronology Of Attacks And Weaknesses, Christopher Meyer und Jörg Schwenk, 31. Januar 2013, http://eprint.iacr.org/2013/049
[XMLCan_V1.0]	Exclusive XML Canonicalization, Version 1.0, W3C Recommendation 18 July 2002, http://www.w3.org/TR/xml-exc-c14n/

[XMLDSig]	XML Signature Syntax and Processing (Second Edition), W3C Recommendation 10 June 2008, http://www.w3.org/TR/2008/PER-xmldsig-core-20080326/
[XMLDSig-Draft]	XML Signature Syntax and Processing Version 2.0, W3C Editor's Draft 04 February 2014, http://www.w3.org/2008/xmlsec/Drafts/xmldsig-core-20/
[XMLDSig-RSA-PSS]	RSA-PSS in XMLDSig, 25/26 September 2007, Konrad Lanz, Dieter Bratko, Peter Lipp, http://www.w3.org/2007/xmlsec/ws/papers/08-lanz-iaik/
[XMLnc]	XML Encryption Syntax and Processing, W3C Recommendation 10 December 2002, http://www.w3.org/TR/xmlenc-core/
[XMLEnc-CM]	Technical Analysis of Countermeasures against Attack on XML Encryption - or - Just Another Motivation for Authenticated Encryption. Juraj Somorovsky, Jörg Schwenk. 2011 http://www.w3.org/2008/xmlsec/papers/xmlEncCountermeasuresW3C.pdf
[XMLEnc-1.1]	XML Encryption Syntax and Processing, W3C Recommendation 11 April 2013, http://www.w3.org/TR/xmlenc-core1/
[XSpRES]	XML Spoofing Resistant Electronic Signature (XSpRES) -- Sichere Implementierung für XML-Signaturen Bundesamt für Sicherheit in der Informationstechnik 2012 https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/SOA/XSpRESS.pfd?__blob=publicationFile
[XSW-Attack]	On Breaking SAML: Be Whoever You Want to Be Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, Meiko Jensen, Usenix 2012 http://www.nds.rub.de/media/nds/veroeffentlichungen/2012/08/03/BreakingSAML.pdf
[Vaudenay-2002]	Security Flaws Induced by CBC Padding: Applications to SSL, IPsec, WTLS … , Serge Vaudenay, Eurocrypt 2002, LNCS 2332/2002, 535-545 https://www.iacr.org/cryptodb/data/paper.php?pubkey=2850