Bundesamt
für Sicherheit in der
Informationstechnik

Technische Richtlinie BSI TR-03116 Kryptographische Vorgaben für Projekte der Bundesregierung

Teil 4: Kommunikationsverfahren in Anwendungen

Stand 2020

Datum: 10. Januar 2020

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
E-Mail: eid@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Technische Richtlinie BSI TR-03116 stellt eine Vorgabe für Projekte des Bundes dar. Die Technische Richtlinie ist in fünf Teile gegliedert:

- Teil 1 der Technischen Richtlinie beschreibt die Sicherheitsanforderungen für den Einsatz kryptographischer Verfahren im Gesundheitswesen für die elektronische Gesundheitskarte (eGK), den Heilberufeausweis (HBA) und der technischen Komponenten der Telematikinfrastruktur.
- Teil 2 beschreibt die Sicherheitsanforderungen für den Einsatz kryptographischer Verfahren in hoheitlichen Dokumenten und eID-Karten basierend auf Extended Access Control, zur Zeit für den elektronischen Reisepass, den elektronischen Personalausweis, den elektronischen Aufenthaltstitel, die eID-Karte für Unionsbürger und den Ankunftsnachweis.
- Teil 3 der Technischen Richtlinie beschreibt die Sicherheitsanforderungen für den Einsatz kryptographischer Verfahren in der Infrastruktur intelligenter Messsysteme im Energiesektor.
- Der vorliegende Teil 4 der Technischen Richtlinie beschreibt die Sicherheitsanforderungen für den Einsatz der Kommunikationsverfahren SSL/TLS, S/MIME, SAML/XML Security und OpenPGP in Anwendungen des Bundes.
- Teil 5 der Technischen Richtlinie beschreibt die Sicherheitsanforderungen für den Einsatz kryptographischer Verfahren in Anwendungen der Secure Element API (wie Technischen Sicherheitseinrichtungen elektronischer Aufzeichnungssysteme).

Die Vorgaben der Technischen Richtlinie basieren auf Prognosen (vgl. [1]) über die Sicherheit der verwendeten kryptographischen Verfahren und Schlüssellängen über einen Zeitraum von 7 Jahren, zur Zeit bis einschließlich 2026. Eine weitere Verwendung des Verfahrens über diesen Zeitraum hinaus ist nicht ausgeschlossen und wird mit 2026+ gekennzeichnet.

Diese Richtlinie macht Vorgaben für die Verwendung von Kommunikationsverfahren. Dabei wird zwischen der Sicherung der eigentlichen Kommunikation und der Zuordnung einer Identität zu den Teilnehmern der Kommunikation unterschieden:

- Abschnitt 2 macht Vorgaben für die Absicherung der Kommunikation mittels TLS.
- Abschnitt 3 macht Vorgaben für die Absicherung von E-Mail-Kommunikation mittels S/MIME.
- Abschnitt 4 macht Vorgaben für die Absicherung der Kommunikation mittels SAML.
- Abschnitt 5 macht Vorgaben für die Identifizierung von Kommunikationspartnern. Ob eine Identifizierung/Authentisierung eines oder beider Partner im konkreten Anwendungsfall notwendig ist, wird durch den Anwendungskontext vorgegeben.
- Anhang A macht Vorgaben für die Absicherung von E-Mail-Kommunikation mittels OpenPGP.

2 Vorgaben für SSL/TLS

Transport Layer Security (TLS), früher bekannt als Secure Socket Layer (SSL), dient der Absicherung der Kommunikation im Internet, z.B. in Verbindung mit HTTP (HTTPS) oder FTP (FTPS). Dabei wird eine sichere Verbindung zwischen zwei Rechnern, dem *Client* und dem *Server*, ausgehandelt und aufgebaut.

Während des Verbindungsaufbaus (*Handshake*) handeln die beiden Parteien die für die nachfolgende Sitzung zu verwendenden Verschlüsselungs- und Authentisierungsalgorithmen (zusammen die *Cipher Suite*) sowie die zu verwendenden Schlüssel selbst aus.

Als weiterer Bestandteil des Handshakes kann eine zertifikatsbasierte Authentisierung eines oder beider Partner bei der Gegenstelle erforderlich sein.

2.1 Allgemeine Vorgaben

Bei der Verwendung von TLS müssen grundsätzlich die Vorgaben und Empfehlungen aus Kapitel 3 der Technischen Richtlinie BSI TR-02102-2 [2] eingehalten werden. Sofern Abweichungen von den Empfehlungen der TR-02102-2 möglich sind, werden diese im vorliegenden Dokument explizit genannt.

Für die eingesetzten kryptographischen Schlüssel und Zufallszahlen gelten die Empfehlungen aus Kapitel 6 dieser Technischen Richtlinie.

In begründeten Ausnahmefällen kann in speziellen Anwendungsszenarien in Abstimmung mit dem BSI von einzelnen Vorgaben aus diesem Kapitel abgewichen werden, sofern diese für die Interoperabilität notwendig sind und hierdurch keine Einschränkungen für das angestrebte Sicherheitsniveau entstehen.

2.1.1 TLS-Versionen und Sessions

Für die Konformität zu dieser Technischen Richtlinie muss mindestens die TLS-Version 1.2 [3] unterstützt werden. Die Unterstützung der TLS-Version 1.3 [4] wird empfohlen. Bei einem Handshake zwischen zu dieser Technischen Richtlinie konformen Clients und Servern muss stets eine dieser TLS-Versionen ausgehandelt werden.

Eine TLS-Session darf eine Lebensdauer von 2 Tagen nicht überschreiten. Dies gilt auch bei der Verwendung von *Session-Resumption*.

2.2 Vorgaben für TLS 1.2

2.2.1 Cipher Suites

Bei TLS 1.2 [3] definiert eine Cipher Suite die zu verwendenden Algorithmen für

- Schlüsseleinigung,
- Verschlüsselung der Datenpakete (Stromchiffre/Blockchiffre inkl. Betriebsmodus) und
- Hashfunktion für die Verwendung im HMAC-Algorithmus für die Integritätssicherung der Datenpakete und für die Verwendung als Pseudozufallszahlengenerator (ab TLS 1.2).

Eine vollständige Liste aller definierten Cipher Suites mit Verweisen auf die jeweiligen Spezifikationen ist verfügbar unter [5].

2.2.1.1 TLS-Clients

Tabelle 1 gibt die von Clients mindestens zu unterstützenden Cipher Suites verbindlich vor. Zudem sollten von TLS-Clients weitere in [2] empfohlene Cipher Suites unterstützt werden.

Cipher Suites	*Zu unterstützen ab*	*Zu unterstützen bis*
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	2013	2026+
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	2013	2026+
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	2015	2026+
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	2015	2026+

Tabelle 1: Von TLS-Clients mindestens zu unterstützende Cipher Suites

Sofern Cipher Suites unterstützt werden, deren Unterstützung sich aus Übergangsregelungen gemäß [2], Kap. 3.3.1.4 ergibt, so müssen diese Cipher Suites von TLS-Clients im Client-Hello mit geringerer Priorität angeboten werden als in [2] regulär empfohlene Cipher Suites.

2.2.1.2 TLS-Server

TLS-Server müssen mindestens ein Zertifikat besitzen, das einen öffentlichen Schlüssel für ECDSA oder RSA enthält. Sofern ein TLS-Server nicht zwei Zertifikate, d.h. für jeden Schlüsseltyp eines, besitzt, wird die Verwendung von ECDSA-Schlüsseln empfohlen[1].

TLS-Server müssen mindestens eine der in Tabelle 2 genannten Cipher Suites verbindlich unterstützen. Zudem sollten serverseitig jeweils weitere in [2] empfohlene Cipher Suites unterstützt werden.

Cipher Suites	*Zu unterstützen ab*	*Zu unterstützen bis*
TLS-Server mit ECDSA-Zertifikat		
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	2013	2026+
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	2015	2026+
TLS-Server mit RSA-Zertifikat		
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	2013	2026+
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	2015	2026+

Tabelle 2: Von TLS-Servern mindestens zu unterstützende Cipher Suites

Sofern Cipher Suites unterstützt werden, deren Unterstützung sich aus Übergangsregelungen gemäß [2], Kap. 3.3.1.4 ergibt, so müssen diese Cipher Suites von TLS-Servern mit geringerer Priorität verwendet werden als in [2] regulär empfohlene Cipher Suites.

1 Vgl. auch Kap. 5.1.

2.2.1.3 Sonderfälle

Sofern anwendungsbezogen Cipher Suites eingesetzt werden, bei denen zusätzlich zur Authentisierung des TLS-Servers via Zertifikaten vorab ausgetauschte Daten (Pre-Shared-Key; PSK) in die Authentisierung und Schlüsseleinigung einfließen, müssen mindestens die Cipher Suites aus Tabelle 3 unterstützt werden.

Cipher Suites	*Verwendung bis*
TLS_RSA_PSK_WITH_AES_128_CBC_SHA256	2026+

Tabelle 3: Cipher Suites basierend auf Zertifikaten und Pre-Shard-Key

Sofern anwendungsbezogen rein PSK-basierte Cipher Suites eingesetzt werden, müssen mindestens die Cipher Suites aus Tabelle 4 unterstützt werden.

Cipher Suites	*Verwendung bis*
TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256	2026+

Tabelle 4: Cipher Suites mit Pre Shared Key

Es wird empfohlen, mindestens serverseitig weitere für den jeweiligen Anwendungsfall empfohlene Cipher Suites aus [2] zu unterstützen.

Sofern Cipher Suites unterstützt werden, deren Unterstützung sich aus Übergangsregelungen gemäß [2], Kap. 3.3.1.4 ergibt, so müssen diese Cipher Suites von Clients und Servern mit geringerer Priorität verwendet werden als in [2] regulär empfohlene Cipher Suites.

2.2.2 Domainparameter

Im Falle von elliptischen Kurven dürfen nur *named curves* (siehe [5]) eingesetzt werden, um Angriffe über nicht verifizierte schwache Domainparameter zu verhindern.

Brainpool-Kurven sollten mit der höchsten Priorität verwendet werden.

Tabelle 5 gibt die mindestens zu unterstützenden elliptischen Kurven verbindlich vor.

Elliptische Kurven	*Zu unterstützen ab*	*Zu unterstützen bis*
secp256r1 (IANA-Nr. 23)	2015	2026+
brainpoolP256r1 (vgl [6], IANA-Nr. 26)	2016	2026+

Tabelle 5: Mindestens zu unterstützende elliptische Kurven

Zudem ist es empfehlenswert, mindestens serverseitig weitere in [2] empfohlene elliptische Kurven zu unterstützen. Abweichend von der TR-02102-2 darf bis einschließlich 2021 auch die elliptische Kurve secp224r1 (IANA-Nr. 22) eingesetzt werden, sofern die weiteren Empfehlungen zu TLS der TR-02102-2 eingehalten werden.

TLS-Clients müssen die Supported-Groups- bzw. Supported-Elliptic-Curves-Extension[2] verwenden, um die unterstützten elliptischen Kurven dem Server mitzuteilen. Im Falle der Unterstützung von DHE-basierten Cipher Suites wird ebenso die Verwendung der Supported-Groups-Extension gemäß [7] empfohlen.

2 Die Supported-Elliptic-Curves-Extension wurde mit [7] in Supported-Groups-Extension umbenannt.

Sowohl TLS-Clients als auch -Server müssen die Verwendung von Domainparametern ablehnen, wenn diese nicht den Anforderungen dieser Technischen Richtlinie entsprechen.

2.2.3 Weitere Vorgaben

2.2.3.1 Signaturalgorithmen

TLS-Clients müssen die Signature-Algorithm-Extension verwenden, um die für die Signaturverifikation unterstützten Paare von Signatur-/Hashalgorithmen anzuzeigen. Erfolgt anwendungsbezogen auch eine Authentisierung des TLS-Clients, so gibt der TLS-Server die von ihm unterstützten Algorithmen in der CertificateRequest-Nachricht an.

Tabelle 6 gibt die jeweils mindestens zu unterstützenden Algorithmen verbindlich vor.

Signaturalgorithmus	***Hashfunktionen***	***Zu unterstützen ab***	***Zu unterstützen bis***
ECDSA[3]	SHA-256	2015	2026+
RSA	SHA-256	2015	2026+

Tabelle 6: Mindestens zu unterstützende Signaturalgorithmen

Abweichend von der TR-02102-2 darf bei den Signaturalgorithmen bis einschließlich 2021 auch die Hashfunktion SHA-224 eingesetzt werden.

2.2.3.2 Encrypt-then-MAC-Extension

Gemäß [3] werden die Klartextdaten bei TLS zunächst integritätsgesichert (MAC) und anschließend werden Klartext und MAC verschlüsselt (MAC-then-Encrypt). Dies führt in Zusammenhang mit einem nicht gesicherten Padding zu Orakelangriffen [8].

Grundsätzlich ist aber die Verwendung von Encrypt-then-MAC oder Authenticated Encryption vorzuziehen [9]. Bei Encrypt-then-MAC werden die zu übertragenen Daten zuerst verschlüsselt und dann MAC gesichert. Daher wird die Verwendung der Encrypt-then-MAC-Extension gemäß [10] empfohlen, d.h. Clients sollten die Encrypt-then-MAC-Extension im Client-Hello anbieten und Server sollten entweder eine GCM-Cipher Suite auswählen oder die Encrypt-then-MAC-Extension im Server-Hello verwenden. Wird die Encrypt-then-MAC-Extension von der jeweils verwendeten TLS-Implementierung unterstützt, so muss diese gemäß [10] angeboten und verwendet werden.

2.2.3.3 OCSP-Stapling

Bei TLS ist insbesondere eine Verifikation des Serverzertifikats erforderlich (vgl. Kap 5.1.3). Grundsätzlich gibt es verschiedene Möglichkeiten Sperrinformationen von Zertifikaten abzufragen. Dabei kann die Abfrage von Rückrufinformationen via OCSP zu einem erhöhten Verbindungsaufkommen bei der zugehörigen CA führen als auch ein Datenschutzproblem für den Client darstellen.

OSCP-Stapling ist eine Methode, bei der Rückrufinformationen in Form von signierten zeitgestempelten OCSP-Antworten dem Client direkt vom Server während des Handshakes bereitgestellt werden.

3 Entfällt bei der Verwendung von RSA_PSK_* Cipher Suites.

Die Verwendung von OCSP-Stapling gemäß [11] (bzw. [12]) wird empfohlen.

2.2.3.4 Session Hash und Extended Master Secret Extension

Im Allgemeinen erfolgt beim TLS-Handshake gemäß [3] die Berechnung des *Master Secrets* so, dass nicht alle kryptographischen Parameter aus dem TLS-Handshake in die Berechnung einbezogen werden. Je nach verwendeten kryptographischen Parametern kann die fehlende Einbeziehung dieser Daten zu Angriffen auf eine TLS-Session führen (vgl. etwa Triple-Handshake-Angriff [13]).

Auch grundsätzlich wird empfohlen, kontextspezifische Daten in die Berechnung von Session-Schlüsseln einzubeziehen. Daher wird die Verwendung der Extended Master Secret Extension gemäß [14] empfohlen. Hierbei fließen die kryptographischen Parameter in Form eines *Session Hashs* (Hashwert über alle Nachrichten des TLS-Handshakes) in die Berechnung des Master Secrets ein.

2.3 Vorgaben für TLS 1.3

2.3.1 Handshake Modi

Bei der Verwendung von TLS 1.3 [4] muss folgender Handshake-Modus unterstützt werden:

- (EC)DHE

Zur Unterstützung von Session Resumption kann zudem folgender Handshake-Modus unterstützt werden:

- PSK mit (EC)DHE

Das Senden oder Annehmen von 0-RTT Daten darf nicht erfolgen.

2.3.2 Cipher Suites

Bei TLS 1.3 definiert eine Cipher Suite die zu verwendenden Algorithmen für

- die authentisierte Verschlüsselung der Datenpakete (Blockchiffre inkl. Betriebsmodus) und
- die Hashfunktion für die Schlüsselableitung.

Eine vollständige Liste aller definierten Cipher Suites mit Verweisen auf die jeweiligen Spezifikationen ist verfügbar unter [5].

Bei der Verwendung von TLS 1.3 sollten von TLS-Clients und -Server mindestens die in Tabelle 7 enthaltenen Cipher Suites unterstützt werden.

Cipher Suites	***Verwendung bis***
`TLS_AES_128_GCM_SHA256`	2026+

Tabelle 7: Von TLS-Clients mindestens zu unterstützende Cipher Suites

Es wird empfohlen, mindestens serverseitig weitere für den jeweiligen Anwendungsfall empfohlene Cipher Suites aus [2] zu unterstützen.

2.3.3 Domain-Parameter

Im Falle von elliptischen Kurven müssen *named curves* (siehe [5]) eingesetzt werden, um Angriffe über nicht verifizierte schwache Domainparameter zu verhindern.

Bei der Verwendung von TLS 1.3 müssen *named curves* (siehe [5]) eingesetzt werden. TLS-Clients und -Server sollten mindestens die in Tabelle 8 enthaltenen elliptischen Kurven unterstützen.

Elliptische Kurven	*Verwendung bis*
secp256r1 (IANA-Nr. 23)	2026+
brainpoolP256r1tls13 (IANA-Nr. 31)	2026+

Tabelle 8: Mindestens zu unterstützende elliptische Kurven

Zudem ist es empfehlenswert, mindestens serverseitig weitere in [2] empfohlene elliptische Kurven zu unterstützen.

2.3.4 Signaturalgorithmen

2.3.4.1 Signaturalgorithmen für den Handshake

Bei der Verwendung von TLS 1.3 müssen TLS-Clients die Signature-Algorithm-Extension verwenden, um die unterstützten Signaturalgorithmen für die Verifikation von Serversignaturen im Rahmen des Handshakes anzuzeigen.

Hierbei sollten von TLS-Clients und -Server mindestens die in Tabelle 9 enthaltenen Signaturalgorithmen unterstützt werden.

Algorithmus	*Verwendung bis*
`ecdsa_secp256r1_sha256`	2026+
`ecdsa_brainpoolP256r1tls13_sha256`	2026+
`rsa_pss_rsae_sha256`	2026+
`rsa_pss_pss_sha256`	2026+

Tabelle 9: Mindestens zu unterstützende Signaturalgorithmen

2.3.4.2 Signaturalgorithmen für die Zertifikatsverifikation

Bei der Verwendung von TLS 1.3 müssen TLS-Clients die Signature-Algorithm-Cert-Extension verwenden, um die für die Zertifikatsverifikation unterstützten Signaturalgorithmen anzuzeigen.

Hierbei sollten von TLS-Clients und -Server mindestens die in Tabelle 9 enthaltenen Signaturalgorithmen für die Zertifikatsverifikation unterstützt werden.

Algorithmus	*Verwendung bis*
rsa_pkcs1_sha256	2025
rsa_pkcs1_sha384	2025
rsa_pss_rsae_sha256	2026+
rsa_pss_pss_sha256	2026+
rsa_pss_rsae_sha384	2026+
ecdsa_secp256r1_sha256	2026+
ecdsa_brainpoolP256r1tls13_sha256	2026+
ecdsa_secp384r1_sha384	2026+
ecdsa_brainpoolP384r1tls13_sha384	2026+

Tabelle 10: Signaturalgorithmen für die Zertifikatsverifikation

3 Vorgaben für S/MIME

Secure/Multipurpose Internet Mail Extensions (S/MIME) sind ein Standard der IETF zur kryptographischen Absicherung von MIME-Nachrichten, wie etwa E-Mails. Hiermit können MIME-Nachrichten digital signiert, verschlüsselt oder komprimiert werden.

Hierbei werden öffentliche Schlüssel verwendet, um Daten zu verschlüsseln bzw. Signaturen zu prüfen. Der zugehörige private Schlüssel dient dazu, verschlüsselte Daten wieder zu entschlüsseln bzw. Signaturen zu erstellen. Der private Schlüssel ist geheim und muss durch geeignete Maßnahmen wie einem Passwort, das nur dem Inhaber des Schlüssels bekannt ist, vor unberechtigtem Zugriff geschützt werden.

S/MIME basiert auf dem CMS-Standard [15] und verwendet als Container für die zu sichernden Daten die CMS-Datenstrukturen SignedData, EnvelopedData bzw. CompressedData.

Die Authentifizierung der Kommunikationspartner erfolgt bei S/MIME PKI-basiert über X.509-Zertifikate. Diese müssen von einer Zertifizierungsstelle ausgestellt werden, um Vertrauen der Kommunikationspartner in die Zertifikate sicherzustellen. Hierbei sind die Vorgaben aus Kap. 5 einzuhalten.

3.1 Versionen

S/MIME wird in mehreren Teilen und Versionen spezifiziert:

- Es wird die Verwendung von S/MIME 4.0 empfohlen. RFC 8551 [16] spezifiziert das Nachrichtenformat und RFC 8550 [17] das Zertifikatshandling von S/MIME 4.0.
- S/MIME 3.2 ([18], [19]) oder 3.1 ([20], [21]) können in bestehenden Anwendungen weiter eingesetzt werden, sofern die Anforderungen dieser Technischen Richtlinie an die zu verwendende Kryptographie eingehalten werden.
- Andere S/MIME-Versionen, als die oben genannten, dürfen nicht verwendet werden.

3.2 Hashfunktionen

S/MIME verwendet Hashfunktionen insbesondere bei der Signierung von Nachrichten sowie bei der Ableitung von Schlüsseln.

Dabei muss eine Hashfunktion aus Tabelle 11 verwendet werden.

Verfahren	*Minimale Outputlänge*	*Verwendung bis*
SHA-2 [22]	224	2022
	256	2026+
	512	2026+

Tabelle 11: Hashfunktionen bei S/MIME

3.3 Signaturen

Für die Erstellung von Signaturen mit S/MIME muss eines der Signaturverfahren aus Tabelle 12 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
RSASSA-PSS [23]	2048	2022
	3072	2026+
ECDSA [24], [22]	224	2022
	256	2026+

Tabelle 12: Signaturverfahren bei S/MIME

3.4 Verschlüsselung

Zur Verschlüsselung von Nachrichten verwendet S/MIME ein hybrides Krypto-System. Die Verschlüsselung der eigentlichen Datenpakete (*Content Encryption*) erfolgt mit einem symmetrischen Verschlüsselungsverfahren. Der zugehörige Schlüssel (*Session Key*) wird zufällig erzeugt und der öffentliche Schlüssel des Empfängers wird dazu verwendet, die Session Keys zu verschlüsseln (*Key Encryption*).

3.4.1 Content Encryption

Für die Content Encryption müssen Verfahren aus Tabelle 13 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
AES CBC-Mode [25]	128	2025
AES GCM-Mode [16]	128	2026+

Tabelle 13: Content Encryption bei S/MIME

3.4.2 Key Encryption

Abhängig von der verwendeten Kryptographie wird der Content Encryption Key direkt mit dem öffentlichen Schlüssel des Empfängers asymmetrisch verschlüsselt (Key Transport) oder der Sender erzeugt ein ephemeres Schlüsselpaar und leitet aus diesem und dem öffentlichen Schlüssel des Empfängers einen symmetrischen Schlüssel ab (Key Agreement), mit dem der Content Encryption Key dann symmetrisch verschlüsselt wird.

Für die asymmetrische Key Encryption muss ein Verfahren aus Tabelle 14 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
RSAES-OAEP [26]	2048	2022
	3072	2026+

Tabelle 14: Asymmetrische Key Encryption bei S/MIME

Für die Key Encryption via Schlüsseleinigung muss ein Verfahren aus Tabelle 15 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
Schlüsselaushandlung		
ECDH [24]	224	2022
	256	2026+
Key-Wrap-Algorithmus		
AES-Wrap [25]	128	2026+

Tabelle 15: Key Encryption via Schlüsseleinigung bei S/MIME

Bei der Ableitung des symmetrischen Schlüssels für die Key Encryption via DH oder ECDH muss eine zulässige Hashfunktion aus Tabelle 11 verwendet werden. Zudem sollten zusätzliche ephemere Daten in die Schlüsselableitung mit einfließen.

3.5 Elliptische Kurven

Bei der Verwendung von elliptischen Kurven dürfen nur Named Curves eingesetzt werden, um Angriffe über nicht verifizierbare schwache Domainparameter zu verhindern. Die folgenden Named Curves sollen verwendet werden.

- BrainpoolP224r1 (bis 2022), BrainpoolP256r1, BrainpoolP384r1, BrainpoolP512r1 (vgl. [27]);
- NIST Curve P-224 (bis 2022), NIST Curve P-256, NIST Curve P-384, NIST Curve P-521.

Es wird die Verwendung der Brainpool-Kurven empfohlen.

3.6 Weitere Vorgaben

- Implementierungen der RSA-Verschlüsselung können relativ leicht Fehler aufweisen, die sie anfällig gegen Chosen-Ciphertext-Angriffe machen(vgl. [28]). Eine Implementierung muss daher geeignete Gegenmaßnahmen vorsehen, so dass solche Angriffe in der Praxis nicht möglich sind (vgl. [26]).
- Grundsätzlich ist es daher eine authentisierte Verschlüsslung sinnvoll. Für die Content Encryption werden von S/MIME bisher noch überwiegend symmetrische Verschlüsselungsalgorithmen ohne Integritätssicherung genutzt. Zudem müssen S/MIME-Implementierungen beliebige Verschachtelungen von Signatur- und Verschlüsselungscontainern unterstützen. Diese Eigenschaften sind grundsätzlich anfällig für Chosen Ciphertext-Attacken bzw. verwandte Angriffe (vgl. etwa [29]).
  Daher müssen geeignete Maßnahmen getroffen werden, um solche Angriffe auf S/MIME-Implementierungen zu verhindern. Insbesondere sollten keine aktiven Inhalte verwendet oder ausgeführt werden. Zudem kann es sinnvoll sein, zusätzliche Sicherheitsmaßnahmen auf Transportebene vorzusehen, um Chosen-Ciphertext-Injection zu verhindern bzw .aufzudecken, vgl. etwa [30] (TLS und DNSSEC/DANE).
- Eine S/MIME-Implementierung sollte bei Erhalt einer signierten bzw. bei Versendung einer verschlüsselten S/MIME-Nachricht mit der Validierung der verwendeten Zertifikate beginnen, um Denial-Of-Service-Angriffe aufgrund von ungültigen Schlüsseln mit extrem hohen Schlüssellängen zu verhindern.

3.7 Mindestanforderungen an die Interoperabilität

Für die Konformität zu dieser Technischen Richtlinie müssen mindestens die folgenden Verfahren unterstützt werden:

- Hashfunktion: SHA-256
- Signaturverfahren: RSASSA-PSS, ECDSA
- Asymmetrische Verschlüsselung: RSAES-OAEP, ECDH
- Symmetrische Verschlüsselung: AES-128 (CBC-Mode)

Außerdem muss die elliptische Kurve BrainpoolP256r1 für die ECC-Verfahren unterstützt werden.

3.8 Übergangsregelungen

Abweichend zu obigen Vorgaben können in bestehenden Anwendungen in begründeten Ausnahmefällen folgende Übergangsregelungen der Tabelle 16 akzeptiert werden.

Abweichung	***Verwendung maximal bis***	***Empfehlung***
Signatur		
RSASSA-PKCS1-v1_5 [31], [22]	2020	Migration auf RSASSA-PSS bzw. ECDSA
Verschlüsselung (Key Encryption)		
RSAES-PKCS1-v1_5 [31], [22]	2015	Migration auf RSAES-OAEP bzw. ECDH

Tabelle 16: Übergangsregelungen für S/MIME

Im Falle der Verwendung müssen geeignete Gegenmaßnahmen gegen Chosen-Ciphertext-Angriffe vorgesehen werden (vgl. [32], [28], [26]).

Für die Verschlüsselung von Nachrichten und die Verifikation von Signaturen können diese Verfahren in bestehenden Anwendungen unterstützt werden, sofern dies für die Interoperabilität mit existierenden nicht-konformen Kommunikationspartnern notwendig ist.

Unabhängig von der angegebenen *maximalen* Verwendung wird eine schnellstmögliche Migration empfohlen.

4 Vorgaben für SAML/XML Security

Extensible Markup Language (XML) ist ein Standard der W3C zur Darstellung hierarchisch strukturierter Daten, welcher in Anwendungen wie z.B. SOAP (Simple Object Access Protocol) oder SAML (Security Assertion Markup Language) genutzt wird.

Die kryptographische Sicherung von XML-Nachrichten bzw. -Nachrichtenteilen basiert auf XML Security (XML Signature und XML Encryption). XML Security ermöglicht die sichere Identifizierung der Kommunikationspartner und erlaubt es,

- die Vertraulichkeit sowie
- die Authentizität und Integrität

von Nachrichten zu sichern. Mit XML Security können sowohl einzelne Nachrichteninhalte als auch ganze Nachrichten gesichert werden. Hierbei wird von der Anwendung bestimmt, welche Inhalte zu verschlüsseln bzw. zu authentisieren sind .

Die Identifizierung der Kommunikationspartner erfolgt in der Regel via X.509-Zertifikaten. Der vertrauenswürdige Austausch kann hierbei PKI-basiert via Zertifikatsketten bzw. signierte Metadaten oder durch bilateralen Schlüsselaustausch erfolgen. Hierbei sind die Vorgaben aus Kap. 5 zu beachten.

4.1 Versionen

Für die Signatur ist hierbei XML Signature gemäß [33] und für die Verschlüsselung ist XML Encryption gemäß [34] zu verwenden. Die im einzelnen zu unterstützenden bzw. zu verwendenden Verfahren werden im Folgenden festgelegt.

4.2 Hashfunktionen

Bei XML Security werden Hashfunktionen für verschiedene Zwecke, wie etwa Signaturen oder Schlüsselableitung, eingesetzt. Dabei muss eine Hashfunktion aus Tabelle 17 verwendet werden.

Verfahren	*Minimale Outputlänge*	*Verwendung bis*
SHA-2 [22]	224	2022
	256	2026+

Tabelle 17: Hashfunktionen bei XML Security

4.3 XML Signature

4.3.1 Signaturen

Für die Signatur von Daten bei XML Security muss eines der Signaturverfahren aus Tabelle 18 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
RSASSA-PSS [26]	2048	2022
	3072	2026+
ECDSA [24], [22]	224	2022
	256	2026+

Tabelle 18: Signaturverfahren bei XML Security

4.4 XML Encryption

Werden Daten bei der Übertragung via XML Security verschlüsselt, so hat dies durch ein hybrides Krypto-System zu erfolgen. Hierbei wird analog zu Kapitel 3.4 der öffentliche Schlüssel des Empfängers dazu genutzt, die Session Keys zu verschlüsseln (*Key Encryption*), und die Verschlüsselung der eigentlichen Datenpakete (*Content Encryption*) erfolgt via symmetrischer Verschlüsselungsverfahren. Der zugehörige Schlüssel für die Content Encryption muss hierbei für jede Übertragung zufällig erzeugt werden.

4.4.1 Content Encryption

Für die Content Encryption müssen Verfahren aus Tabelle 19 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
AES GCM-Mode [34]	128	2026+

Tabelle 19: Content Encryption bei XML Security

4.4.2 Key Encryption

Die Key Encryption kann per Schlüsseltransport (Key Transport) oder per Schlüsselableitung (Key Agreement) umgesetzt werden.

4.4.2.1 Key Transport

Beim Key Transport muss ein Verfahren aus Tabelle 20 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
RSAES-OAEP [34]	2048	2022
	3072	2026+

Tabelle 20: Key Transport bei XML Security

4.4.2.2 Key Agreement

Beim Key Agreement muss ein Verfahren aus Tabelle 21 verwendet werden. Hierbei muss für die Schlüsselableitung eine Hashfunktion gemäß Kapitel 4.2 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
Schlüsselaushandlung		
ECDH [34]	224	2022
	256	2026+
Key-Wrap-Algorithmus		
AES-Wrap [34]	128	2026+

Tabelle 21: Key Agreement bei XML Security

4.5 Elliptische Kurven

Bei der Verwendung von elliptischen Kurven dürfen nur Named Curves eingesetzt werden, um Angriffe über nicht verifizierbare schwache Domainparameter zu verhindern. Die folgenden Named Curves sollen verwendet werden.

- BrainpoolP224r1, BrainpoolP256r1, BrainpoolP384r1, BrainpoolP512r1 (vgl. [27]);
- NIST Curve P-224, NIST Curve P-256, NIST Curve P-384, NIST Curve P-521.

Es wird die Verwendung der Brainpool-Kurven empfohlen.

4.6 Mindestanforderungen an die Interoperabilität

Für die Konformität zu dieser Technischen Richtlinie müssen mindestens die folgenden Verfahren unterstützt werden:

- Hashfunktion:
  - http://www.w3.org/2001/04/xmlenc#sha256
- Signaturverfahren:
  - http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
- Key Encryption:
  - http://www.w3.org/2009/xmlenc11#ECDH-ES
  - http://www.w3.org/2001/04/xmlenc#kw-aes128
- Key Derivation
  - http://www.w3.org/2009/xmlenc11#ConcatKDF
- Content Encryption:
  - http://www.w3.org/2009/xmlenc11#aes128-gcm

Außerdem muss die elliptische Kurve BrainpoolP256r1 für die ECC-Verfahren unterstützt werden.

4.7 Übergangsregelungen

Abweichend zu obigen Vorgaben können in bestehenden Anwendungen in begründeten Ausnahmefällen folgende Übergangsregelungen der Tabelle 22 akzeptiert werden.

Abweichung	*Verwendung maximal bis*	*Empfehlung*
Signatur		
RSASSA-PKCS1-v1_5	2020	Migration auf RSASSA-PSS bzw. ECDSA
Content Encryption		
AES CBC-Mode	2020	Migration auf AES GCM-Mode

Tabelle 22: Übergangsregelungen für SAML

Im Falle der Verwendung müssen geeignete Gegenmaßnahmen gegen Chosen-Ciphertext-Angriffe vorgesehen werden (vgl. [32], [28], [26]).

Unabhängig von der angegebenen *maximalen* Verwendung wird eine schnellstmögliche Migration empfohlen.

5 Identifizierung von Kommunikationspartnern

Für die sichere Kommunikation ist meist die Identifizierung eines oder mehrerer Beteiligter notwendig. Die in dieser Richtlinie betrachteten Verfahren nutzen die Zuordnung eines asymmetrischen Schlüsselpaares zu einer Entität zur Identifizierung dieser Entität. Die Zuordnung des Schlüsselpaares kann entweder durch Nutzung einer Public-Key-Infrastruktur (siehe 5.1) oder durch direkten bilateralen Austausch von öffentlichen Schlüsseln bzw. Zertifikaten über einen vertrauenswürdigen Kanal (siehe 5.2) erfolgen.

5.1 PKI-basierte Identifizierung

SSL/TLS, S/MIME und XML Security (wie etwa SAML) unterstützen die PKI-basierte Identifizierung eines oder beider Kommunikationspartner. Die hierzu genutzte PKI-Struktur, die *Internet-PKI*, wird in [35] spezifiziert.

5.1.1 Zertifizierungsstellen/Vertrauensanker

Bei Nutzung einer PKI-basierten Identifizierung werden die Zertifikate für die Kommunikationspartner von einer oder mehreren Zertifizierungsstellen (CAs) ausgestellt. Eine Anwendung muss für die Verifikation von Zertifikaten einen oder mehrere Vertrauensanker vorhalten, d.h. Wurzelzertifikate vertrauenswürdiger Zertifizierungsstellen.

Die Auswahl der Zertifizierungsstellen für die Zertifikatsausstellung und die Auswahl der vorgehaltenen Vertrauensanker muss mit großer Sorgfalt erfolgen. Bei der Auswahl sollten insbesondere die folgenden Kriterien berücksichtigt werden:

- IT-Sicherheit des CA-Betriebs, geprüft durch einen Audit/eine Zertifizierung nach einem anerkannten Audit-/Zertifizierungs-Standard;
  - Es wird eine Zertifizierung nach BSI TR-03145 [36] empfohlen;
- Hohes Sicherheitsniveau der Registrierungsservices, einschließlich an Dienstleister (Registrare) ausgelagerten Services;
- Vertrauenswürdigkeit des Betreibers und des Betriebs, auch unter Berücksichtigung von Eingriffsrechten Dritter;
- Verfügbarkeit des Rückrufservice;
- Rechtsstand, insbesondere in Bezug auf das geltende Haftungs- und Datenschutzrecht.

Für Anwendungen, in denen ein hohes Vertrauensniveau erreicht werden soll, müssen Zertifizierungsstellen über ein TR-Zertifikat nach [36] verfügen.

Die Zahl der Vertrauensanker sollte so gering wie möglich gehalten werden.

5.1.2 Zertifikate

Die Zertifikatsstruktur ist in [35] beschrieben, kann aber anwendungsbezogen weiter eingeschränkt bzw. um weitere Extensions ergänzt werden.

Für die Konformität zu dieser Richtlinie müssen Endnutzerzertifikate und CA-Zertifikate für Anwendungen die folgenden Anforderungen erfüllen:

- Alle Zertifikate müssen Informationen für eine Rückrufprüfung enthalten, d.h.
  - einen `CRLDistributionPoint`, unter dem jederzeit aktuelle CRLs zur Verfügung stehen, oder
  - eine `AuthorityInfoAccess`-Extension, welche die notwendigen Informationen zur Abfrage eines OCSP-Servers enthält.
- Alle Zertifikate müssen eine geeignete Gültigkeitsdauer enthalten
  - Endnutzerzertifikate dürfen eine Gültigkeitsdauer von höchstens drei Jahren nicht überschreiten.
  - Sub-CA-Zertifikate dürfen eine Gültigkeitsdauer von höchstens fünf Jahren nicht überschreiten.
  - Vertrauensanker (Wurzelzertifikate) sollten eine Gültigkeitsdauer von maximal 6 Jahren nicht überschreiten. Ist eine Aktualisierung der Vertrauensanker bei den vertrauenden Stellen im Falle einer Kompromittierung/Auslaufen der Eignung der kryptographischen Algorithmen nicht sichergestellt, ist eine geeignete kürzere Gültigkeitszeit zu wählen.
- CA-Zertifikate müssen eine als kritisch markierte `BasicConstraints`-Extension enthalten. In CA-Zertifikaten muss das in der Extension enthaltene Feld `pathLenConstraint` vorhanden sein und auf einen möglichst kleinen Wert gesetzt werden.
- Alle Zertifikate müssen eine `KeyUsage`-Extension enthalten, die die mit dem Zertifikat verbundenen Rechte so weit wie möglich einschränkt und als kritisch markiert ist. Endnutzerzertifikate sollten darüber hinaus eine `ExtendedKeyUsage`-Extension enthalten, die die mit dem Zertifikat verbundenen Rechte so weit wie möglich einschränkt.
- Für verschiedene Anwendungszwecke (Signatur, Verschlüsselung, Authentisierung, usw.) sollten nach Möglichkeit verschiedene Schlüsselpaare generiert und dementsprechend verschiedene Zertifikate ausgestellt und verwendet werden.
- Zertifikate dürfen keine Wildcards im `CommonName` des `Subject` oder `SubjectAltName` enthalten.

Für Browser-basierte Anwendungen (Webseiten) wird die Verwendung von qualifizierten Webseiten-Zertifikaten gemäß [37] bzw. Extended-Validation-Zertifikaten empfohlen, insbesondere wenn im Rahmen der Anwendung personenbezogene Daten verarbeitet werden.

5.1.3 Zertifikatsverifikation

Bei der Überprüfung eines Zertifikats sind die Regeln aus [35], Abschnitt 6 „Certification Path Validation“, vollständig umzusetzen. Dies umfasst insbesondere:

- vollständige Prüfung der Zertifikatskette bis zu einem für die jeweilige Anwendung vertrauenswürdigen und als authentisch bekannten Vertrauensanker (vgl. 5.1.1);
- Prüfung auf Gültigkeit (Ausstellungs- und Ablaufdatum);
- Rückrufprüfung *aller* Zertifikate der Kette;
  - Im Falle von TLS wird hierbei aus Performance- und Datenschutzgründen die Verwendung von OCSP-Stapling empfohlen, vgl. Kap. 2.2.3.3.
- Auswertung der in den Zertifikaten enthaltenen Extensions gemäß den Regeln in [35], insbesondere alle in 5.1.2 vorgegebenen Extensions.

In bestimmten Anwendungen kann von den Vorgaben dieses Abschnittes, begründet und in Abstimmung mit dem BSI, abgewichen werden.

5.1.4 Domainparameter und Schlüssellängen

Die Schlüssel für die Signatur von Zertifikaten müssen mindestens die Anforderungen aus Tabelle 23 erfüllen.

Algorithmus	*Minimale Schlüssellänge*	*Min. Outputlänge der Hashfunktion*	*Verwendung bis*
ECDSA [38]	224 Bit	SHA-224	2021
	256 Bit	SHA-256	2026+
DSA [38]	2048 Bit	SHA-224	2021
	3072 Bit	SHA-256	2026+
RSASSA-PSS [39]	2048 Bit	SHA-224	2021
	3072 Bit	SHA-256	2026+

Tabelle 23: Signaturalgorithmen und Mindestschlüssellängen für X.509-Zertifikate

Es wird empfohlen, für die Signatur von Zertifikaten, Schlüssel mit einer Bitlänge zu verwenden, die mindestens so groß wie die des im Zertifikat enthaltenen Schlüssels ist. Zudem wird empfohlen, für Wurzelzertifikate – soweit möglich – längere Schlüssel als für nachgeordnete Zertifikate bzw. Endnutzerschlüssel zu verwenden.

Bei der Verwendung von elliptischen Kurven (ECC) dürfen nur Named Curves eingesetzt werden, um Angriffe über nicht verifizierbare schwache Domainparameter zu verhindern. Die folgenden Named Curves sollen verwendet werden.

- BrainpoolP224r1[4], BrainpoolP256r1, BrainpoolP384r1, BrainpoolP512r1 (vgl. [27]);
- NIST Curve P-224, NIST Curve P-256, NIST Curve P-384, NIST Curve P-521.

Es wird die Verwendung von ECC mit Brainpool-Kurven empfohlen.

5.1.4.1 Übergangsregelungen

Abweichend zu obigen Vorgaben können in bestehenden Anwendungen für die Erzeugung von Zertifikaten in begründeten Ausnahmefällen folgende Übergangsregelungen der Tabelle 24 mit den entsprechenden Schlüssellängen der Tabelle 23 verwendet werden.

4 Für diese Kurve existiert beim TLS-Protokoll keine ID.

Abweichung	*Verwendung maximal bis*	*Empfehlung*
RSASSA-PKCS1-v1_5	2020	Migration auf RSASSA-PSS bzw. ECDSA

Tabelle 24: Übergangsregelungen für die Signatur von Zertifikaten

Unabhängig von der angegebenen *maximalen* Verwendung wird eine schnellstmögliche Migration empfohlen.

5.2 Identifizierung über bilateralen Schlüsselaustausch bzw. Web of Trust

Nicht auf einer PKI basiert die Identifizierung einer Entität beim Austausch von Vertrauensankern einer PKI und im Web of Trust (z.B. OpenPGP).

Vertrauensanker einer PKI verwenden selbstsignierte Zertifikate, deren Authentizität durch einen vertrauenswürdigen bilateralen Schlüsselaustausch sichergestellt werden muss.

Im Web of Trust verwenden Teilnehmer selbstsignierte Zertifikate, deren Authentizität dezentral durch Signaturen von weiteren Entitäten des Web Of Trusts bestätigt wird. Auch bei SAML können je nach Anwendungsszenario selbstsignierte Zertifikate zum Einsatz kommen.

5.2.1 Identifizierung von Zertifikatsinhabern

Die Identität einer Entität wird in diesen Systemen also nicht zentral von einer Registrierungsstelle einer Zertifizierungsinstanz geprüft, sondern die Übermittlung des Zertifikats muss über einen vertrauenswürdigen Kommunikationskanal erfolgen.

Das Vertrauen in das Zertifikat einer Entität muss hierbei mittels einem der folgenden Verfahren sichergestellt werden.

1. Direkter bilateraler Austausch der selbstsignierten Zertifikate über einen vertrauenswürdigen Kanal. Die einzuhaltenden Sicherheitsanforderungen sind dazu zwischen den Kommunikationspartnern bilateral zu vereinbaren und zu prüfen. Die Sicherheitsanforderungen müssen so gestaltet werden, dass die Kommunikationspartner ein dem Schutzbedarf der Anwendung angemessenes Vertrauen in die Authentizität der Zertifikate erhalten. Beispiele für Anforderungen sind persönlicher Austausch der Zertifikate mit vorhergehender Identifizierung mittels Ausweis oder Abgleich eines Zertifikatsfingerprints auf einem unabhängigen und authentisierten Kanal.
2. (Web Of Trust) Signatur von Zertifikaten durch einen vertrauenswürdigen Dritten. Hierbei muss sowohl die Authentizität des vertrauenswürdigen Dritten als auch die Authentizität des signierten Schlüssels sichergestellt werden. Die jeweiligen Sicherheitsanforderungen sind hierbei durch sämtliche beteiligten Entitäten einzuhalten, auf die sich die Authentifizierung stützt.

Besonderer Wert sollte jeweils auf ein hohes Sicherheitsniveau des Identifizierungsprozesses gelegt werden.

5.2.2 Weitergabe von Zertifikaten

Stellt eine Entität im Web Of Trust einem Kommunikationspartner auch Schlüssel von Dritten zur Verfügung, so muss diese durch Vereinbarungen sicherstellen, dass das erforderliche Sicherheitsniveau der Identifizierung durch sämtliche beteiligten Stellen eingehalten wird.

Möglichkeiten der Veröffentlichung von Zertifikaten sind etwa Schlüsselserver oder Masterlisten.

5.2.3 Rückruf

Der Rückruf von Zertifikaten (*Revocation*) stellt in nicht-PKI-basierten Systemen ein besonderes Problem dar, da nicht sichergestellt ist, dass ein Rückruf – etwa aufgrund einer Schlüsselkompromittierung – unmittelbar allgemein bekannt wird.

Erfolgt der Zertifikatsaustausch bilateral, so muss der Inhaber im Falle eines zurückgerufenen Schlüssels unmittelbar alle direkten Kommunikationspartner über den Rückruf informieren, mit denen ein bilateraler Zertifikatsaustausch stattgefunden hat.

Im Web Of Trust muss für einen zurückgerufenen Schlüssel zusätzlich ein Rückrufzertifikat auf den Schlüsselservern veröffentlicht werden, von denen dem Schlüsselinhaber bekannt ist, dass der jeweilige Schlüssel dort veröffentlicht ist.

Zu zurückgerufenen Zertifikaten gehörende Schlüssel dürfen nicht mehr verwendet werden.

5.2.4 Domainparameter und Schlüssellängen

Bei der Identifizierung via bilateralem Zertifikatsaustausch ergeben sich zu verwendenden Domainparameter und Schlüssellängen aus den Vorgaben an die jeweiligen Signaturschlüssel der Zertifikatsinhaber bzw. der Signaturersteller. Zertifikate dürfen eine Gültigkeitsdauer von maximal 5 Jahren haben.

6 Kryptographische Schlüssel

6.1 Erzeugung

Kryptographische Schlüssel sollten grundsätzlich unter der Kontrolle des Schlüsselinhabers erzeugt werden. Eine Erzeugung eines Schlüssels z.B. bei der zertifikatsausstellenden CA ist nur in begründeten Ausnahmefällen zulässig. In diesem Fall muss sichergestellt werden, dass nach Auslieferung des Schlüssels an den Inhaber keine Kopien bei der erzeugenden Stelle verbleiben und die Auslieferung vertraulich erfolgt.

6.2 Zufallszahlen

Für die Generierung von Zufallszahlen, z.B. für die Erzeugung kryptographischer Schlüssel oder für die Signaturerzeugung, müssen geeignete Zufallszahlengeneratoren eingesetzt werden.

Empfohlen wird ein Zufallszahlengenerator einer der Klassen DRG.3, DRG.4, PTG.3 oder NTG.1 nach [40]. Weitere Informationen über die Erzeugung asymmetrischer Schlüssel sind auch in [1], Anhang B, zu finden.

6.3 Speicherung und Verarbeitung

Private kryptographische Schlüssel, insbesondere statische Schlüssel und Signaturschlüssel, müssen sicher gespeichert und verarbeitet werden. Dies bedeutet u.a. den Schutz vor Kopieren, missbräuchlicher Nutzung und Manipulation der Schlüssel. Eine sichere Schlüsselspeicherung kann z.B. durch die Verwendung entsprechend zertifizierter Hardware (Chipkarte, HSM) gewährleistet werden.

Ebenso müssen die öffentlichen Schlüssel von als vertrauenswürdig erkannten Stellen (Vertrauensanker) sowie bilateral ausgetauschte Schlüssel manipulationssicher gespeichert werden.

6.4 Vernichtung

Private kryptographische Schlüssel, Geheimnisse u.ä. müssen unmittelbar gelöscht werden, sobald sie nicht mehr benötigt werden. Das Löschen muss dabei sicher erfolgen. Ein reines Deaktivieren der Schlüssel reicht i.A. nicht aus.

A Vorgaben für OpenPGP

Pretty Good Privacy (PGP) ist ein System zur kryptographischen Absicherung von Daten, insbesondere von E-Mails und Dateien. Mit PGP können Daten digital signiert und verschlüsselt werden.

Nutzer verwenden hierbei im Allgemeinen öffentliche Schlüssel eines Kommunikationspartners, um diesem verschlüsselte Daten zu übermitteln bzw. dessen Signaturen über empfangene Daten zu prüfen. Die zugehörigen privaten Schlüssel besitzt nur der jeweilige Schlüsselinhaber. Diese dienen zur Entschlüsselung verschlüsselter Daten bzw. Erstellung von Signaturen durch den Schlüsselinhaber und sind in der Regel durch ein Passwort geschützt.

Die Authentifizierung der Kommunikationspartner basiert auf dem Web of Trust. Hierbei sind die Vorgaben aus Kap. 5 einzuhalten. Bei OpenPGP muss die Sicherstellung des Vertrauens in Zertifikate ebenso wie der Rückruf von Zertifikaten durch die Endanwender selbst realisiert werden. Hierfür ist im Web Of Trust ein erhebliches Fachwissen erforderlich und die Einhaltung der notwendigen Anforderungen an die Identifizierung kann nur sehr eingeschränkt geprüft werden. Daher wird der Einsatz von OpenPGP im eGovernment i.A. nicht empfohlen. OpenPGP sollte daher nur verwendet werden, wenn sichergestellt ist, dass dieses Wissen bei allem Beteiligten vorhanden ist und alle Anforderungen an die Identitifizierung erfüllt sind. In diesem Fall sind die Anforderungen aus diesem Anhang einzuhalten.

Es gibt verschiedene, teilweise inkompatible Versionen von PGP. Darunter ist OpenPGP (aufbauend auf PGP 5.x) heute internationaler Internet-Standard [41].

A.1 Versionen

PGP muss in der standardisierten Version OpenPGP konform zu [41] verwendet werden. Es wird die Unterstützung von Elliptischer-Kurven-Kryptographie (ECC) gemäß [42] empfohlen.

Als Nachrichtenformat für signierte oder verschlüsselte E-Mails sollte das PGP/MIME-Format nach [43] verwendet werden. Die Verwendung des PGP/INLINE-Formats wird nicht empfohlen.

A.2 Hashfunktionen

Es muss eine Hashfunktion aus Tabelle 25 verwendet werden.

Verfahren	***Minimale Schlüssel-/Outputlänge***	***Verwendung bis***
SHA-2 [41]	224	2022
	256	2026+

Tabelle 25: Hashfunktionen bei OpenPGP

A.3 Signaturen

Bei der Nutzung von OpenPGP für die Erstellung von Signaturen muss ein Verfahren aus Tabelle 26 verwendet werden.

Verfahren	*Minimale Schlüssel-/Outputlänge*	*Verwendung bis*
RSASSA-PKCS1-v1_5 [41]	2048	2019
DSA [41]	2048	2022
	3072	2026+
ECDSA [42]	256	2026+

Tabelle 26: Signaturverfahren bei OpenPGP

A.4 Verschlüsselung

Bei OpenPGP erfolgt die Verschlüsselung der eigentlichen Datenpakete (*Content Encryption*) via symmetrischer Verschlüsselung, wobei der zugehörige Schlüssel (*Session Key*) zufällig erzeugt und im Allgemeinen mit dem öffentlichen Schlüssel des Empfängers asymmetrisch verschlüsselt (*Session Key Encryption*) wird. Alternativ ist es bei OpenPGP auch möglich die Session Keys mittels vorab ausgehandelter geheimer Daten (Passphrase) symmetrisch zu verschlüsseln.

Sofern anwendungsspezifisch möglich, soll die asymmetrische Key Encryption verwendet werden

A.4.1 Verschlüsselung von Datenpaketen (Content Encryption)

Für die Verschlüsselung der Datenpakete muss ein Verfahren aus Tabelle 27 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
AES CFB-Mode[5] [41]	128	2026+

Tabelle 27: Symmetrische Verschlüsselung von Datenpaketen (Content Encryption) mit OpenPGP

Die Datenpakete sollen durch Verwendung eines Modification Detection Codes (Symmetrically Encrypted Integrity Protected Data Packets) gegen Fälschung geschützt werden.

A.4.2 Asymmetrische Verschlüsselung der Session Keys

Für die asymmetrische Verschlüsselung der Session Keys muss ein Verfahren aus Tabelle 28 verwendet werden.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
RSAES-PKCS#1-v1_5 [41]	2048	2015
ElGamal [41]	2048	2022
	3072	2026+

Tabelle 28: Asymmetrische Verschlüsselung der Session Keys (Session Key Encryption) bei OpenPGP

Ebenso kann die Verschlüsselung via Schlüsseleinigung erfolgen. Hierbei sind die Vorgaben aus Tabelle 29 einzuhalten. Empfehlungen zur Kombination der jeweiligen Schlüssellängen werden in [42] gegeben.

5 OpenPGP verwendet als Betriebsart eine Variante des CFB-Modes (vgl. [41]).

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
Schlüsselaushandlung		
ECDH [42]	256	2026+
Algorithmus für die KEK Encryption		
AES-Wrap [42]	128	2026+

Tabelle 29: Verschlüsselung der Session Keys (Session Key Encryption) bei OpenPGP via Schlüsseleinigung

Bei der Ableitung des symmetrischen Schlüssels mittels ECDH aus dem Shared Secret muss eine zulässige Hashfunktion aus Tabelle 25 verwendet werden.

A.4.3 Symmetrische Verschlüsselung von Session Keys und Schutz eines privaten Schlüssels

Sofern es anwendungsspezifisch nötig ist, können die Session Keys auch mittels vorab ausgehandelter geheimer Daten (Passphrase) verschlüsselt werden. Zudem werden Passphrasen bei OpenPGP verwendet, um den privaten Schlüssel zu schützen. Hierbei sind die Vorgaben aus Tabelle 30 einzuhalten.

Verfahren	*Minimale Schlüssellänge*	*Verwendung bis*
AES CFB-Mode [41]	128	2026+

Tabelle 30: Symmetrische Verschlüsselung von Session Keys bei OpenPGP

In die Ableitung des Schlüssels für die Key Encryption via symmetrischer Verschlüsselung müssen zusätzliche ephemere Daten (Salt Value) einfließen.

Bei der Ableitung des symmetrischen Schlüssels aus der Passphrase muss eine zulässige Hashfunktionen aus Tabelle 25 verwendet werden.

A.5 Elliptische Kurven

Bei der Verwendung von elliptischen Kurven dürfen nur Named Curves eingesetzt werden, um Angriffe über nicht verifizierbare schwache Domainparameter zu verhindern. Die folgenden Named Curves sollen verwendet werden.

- NIST Curve P-256, NIST Curve P-384, NIST Curve P-521.

A.6 Weitere Vorgaben

- Die RSA-Verschlüsselung ist grundsätzlich anfällig gegen Chosen-Ciphertext-Angriffe (vgl. [32], [28]). Eine Implementierung muss daher geeignete Gegenmaßnahmen vorsehen, dass solche Angriffe in der Praxis nicht möglich sind (vgl. [44]).
- Für die Content Encryption werden von OpenPGP symmetrische Verschlüsselungsalgorithmen ohne Integritätssicherung genutzt. Diese Eigenschaften sind grundsätzlich anfällig für Chosen Ciphertext-Attacken bzw. verwandte Angriffe (vgl. etwa [29], [45]). Daher müssen geeignete Maßnahmen getroffen werden, um Chosen-Chiphertext-Attacken auf die symmetrische Verschlüsselung von OpenPGP-Implementierungen zu verhindern. Insbesondere sollten keine aktiven Inhalte verwendet oder ausgeführt werden.

- Eine OpenPGP-Implementierung sollte bei Erhalt einer signierten bzw. bei Versendung einer verschlüsselten PGP-Nachrichten mit der Validierung der verwendeten Zertifikate beginnen, um Denial-Of-Service-Angriffe aufgrund von ungültigen Schlüsseln mit extrem hohen Schlüssellängen zu verhindern.

A.7 Mindestanforderungen an die Interoperabilität

Für die Konformität zu dieser Technischen Richtlinie müssen mindestens die folgenden Verfahren unterstützt werden:

- Hashfunktion: SHA-256
- Signaturverfahren: DSA, ECDSA
- Asymmetrische Verschlüsselung: ElGamal, ECDH
- Symmetrische Verschlüsselung: AES-128

Zudem muss die elliptische Kurve NIST Curve P-256 unterstützt werden.

Literaturverzeichnis

[1] BSI TR-02102-1, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 1, 2020

[2] BSI TR-02102-2, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Teil 2 - Verwendung von Transport Layer Security (TLS), 2020

[3] IETF RFC 5246, T. Dierks, E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2

[4] IETF RFC 8446, E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3

[5] IANA, http://www.iana.org/assignments/tls-parameters/tls-parameters.xml

[6] IETF RFC 7027, J, Merkle, M. Lochter, Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)

[7] IETF RFC 7919, D. Gillmor, Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS), 2016

[8] Nadhem AlFardan, Kenny Paterson, Lucky Thirteen: Breaking the TLS and DTLS Record Protocols, http://www.isg.rhul.ac.uk/tls/

[9] M. Bellare, C. Namprempre, Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm; in Advances in Cryptology - Asiacrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed, Springer-Verlag, 2000.

[10] IETF RFC 7366, P. Gutman, Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS), 2014

[11] IETF RFC 6961, Y. Pettersen, The Transport Layer Security (TLS) Multiple Certificate Status Request Extension, 2013

[12] IETF RFC 6066, D. Eastlake, Transport Layer Security (TLS) Extensions: Extension Definitions, 2011

[13] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, A. Pironti, P.-Y. Strub, Triple Handshake and Cookie Cutters: Breaking and Fixing Authentication over TLS, IEEE Symposium on Security and Privacy, 2014

[14] IETF RFC 7627, K. Bhargavan, Ed., A. Delignat-Lavaud, A. Pironti, A. Langley, M. Ray, Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension

[15] IETF RFC 5652, R. Housley, Cryptographic Message Syntax (CMS), 2009

[16] IETF RFC 8551, J. Schaad, B. Ramsdell, S. Turner, Secure/Multipurpose Internat Mail Extension (S/MIME) Version 4.0 Message Specification

[17] IETF RFC 8550, J. Schaad, B. Ramsdell, S. Turner, Secure/Multipurpose Internat Mail Extension (S/MIME) Version 4.0 Certificate Handling

[18] IETF RFC 5751, B. Ramsdell, S.Turner, Secure/Multipurpose Interenet Mail Extensions (S/MIME) Version 3.2 Message Specification, 2010

[19] IETF RFC 5750, B. Ramsdell, S. Turner, Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling, 2010

[20] IETF RFC 3851, B.Ramsdell, Secure/Multipurpose Mail Extensions (S/MIME) Version 3.1 Message Specification, 2004

[21] IETF RFC 3850, B. Ramsdell, Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 - Certificate Handling, 2004

[22] IETF RFC 5754, S. Turner, Using SHA2 Algorithms with Cryptographic Message Syntax, 2010

[23] IETF RFC 4056, J. Schaad, Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS), 2005

[24] IETF RFC 5753, S.Turner, D. Brown, Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS), 2010

[25] IETF RFC 3565, J.Schaad, Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS), 2003

[26] IETF RFC 8017, K. Moriarty, J. Jonsson, B. Kaliski, A. Rusch, PKCS #1: RSA Cryptography Specifications Version 2.2, 2016

[27] IETF RFC 5639, M. Lochter, J. Merkle, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation
[28] J. Manger, A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0, Advances in Cryptology - Crypto 2001, Lecture Notes in Computer Science, vol. 2139, pp. 260-274, 2001
[29] Jonathan Katz, Bruce Schneier , A Chosen Ciphertext Attack Against Several E-Mail Encryption Protocols, Usenix Security Symposium 2000
[30] BSI TR-03108, Secure E-Mail Transport
[31] IETF RFC 3370, R. Housley, Cryptographic Message Syntax (CMS) Algorithms, 2003
[32] D. Bleichenbacher, Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1, Advances in Cryptology - Crypto '98, Lecture Notes in Computer Science, vol. 1462, pp. 1-12, Springer Verlag, 1998
[33] W3C , XML Signature Syntax and Processing Version 1.1
[34] W3C, XML Encryption Syntax and Processing Version 1.1
[35] IETF RFC 5280, D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
[36] BSI TR-03145, Secure Certification Authority operation
[37] REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCILof 23 July 2014on electronic identification and trust services for electronic transactions in the internal market andrepealing Directive 1999/93/EC
[38] IETF RFC 5758, Q. Dang, S. Santesson, K. Moriarty, D. Brown, T.Polk, Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA, 2010
[39] IETF RFC 4055, J, Schaad, B. Kaliski, R. Housley, Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2005
[40] BSI AIS 20/31, A proposal for: Functionality classes for random number generators
[41] IETF RFC 4880, J, Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer, OpenPGP Message Format, 2007
[42] IETF RFC 6637, A. Jivsov, Elliptic Curve Cryptography (ECC) in OpenPGP, 2012
[43] IETF RFC 3156, M. Elkins, D. Del Torto, R. Levien, T. Rossler, MIME Security with OpenPGP, 2001
[44] IETF RFC 3447, J. Jonsson, B. Kaliski, Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1, 2003
[45] K. Jallal, J. Katz, J. J. Lee, B. Schneier , Implementation of Chosen Ciphertext Attacks against PGP and GnuPGP