Bundesamt
für Sicherheit in der
Informationstechnik

Technische Richtlinie TR-02102-2 Kryptographische Verfahren: Empfehlungen und Schlüssellängen

Teil 2 – Verwendung von Transport Layer Security (TLS)

Version 2020-01

Änderungshistorie

Version	Datum	Beschreibung
2019-01	22.02.2019	Anpassung der Verwendungszeiträume, Empfehlung von TLS 1.3, Empfehlung von PSK-Cipher-Suiten aus RFC 8442, Empfehlung des CCM-Modus
2020-01	28.02.2020	Anpassung der Verwendungszeiträume, Abkündigung von HMAC-SHA-1

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
E-Mail: TR02102@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

Tabellenverzeichnis

1 Einleitung

Diese Technische Richtlinie enthält Empfehlungen für den Einsatz des kryptographischen Protokolls *Transport Layer Security (TLS)*. Es dient der sicheren Übertragung von Informationen in Datennetzwerken, wobei insbesondere die *Vertraulichkeit*, die *Integrität* und die *Authentizität* der übertragenen Informationen im Vordergrund stehen.

Die vorliegende Technische Richtlinie enthält Empfehlungen für die zu verwendende Protokollversion sowie die kryptographischen Algorithmen und Schlüssellängen als Konkretisierung der allgemeinen Empfehlungen in Teil 1 dieser Technischen Richtlinie [TR-02102-1]. Wie in Kapitel 1 von [TR-02102-1] angemerkt, werden nicht aufgeführte kryptographische Verfahren vom BSI nicht unbedingt als unsicher beurteilt.

Diese Technische Richtlinie enthält keine Empfehlungen für konkrete Anwendungen, keine Risikobewertungen sowie keine Angriffsmöglichkeiten, die sich aus Fehlern in der Implementierung des Protokolls ergeben.

Hinweis: Auch bei Beachtung aller Empfehlungen für die Verwendung von TLS können Daten in erheblichem Umfang aus einem kryptographischen System abfließen, zum Beispiel durch Ausnutzung von Seitenkanälen (Messung von Timing-Verhalten, Stromaufnahme, Datenraten etc.). Daher sollte der Entwickler eines kryptographischen Systems unter Hinzuziehung von Experten auf diesem Gebiet mögliche Seitenkanäle identifizieren und entsprechende Gegenmaßnahmen umsetzen. Je nach Anwendung gilt dies auch für Fault-Attacken.

Hinweis: Für Definitionen kryptographischer Begriffe in diesem Dokument siehe das Glossar in [TR-02102-1].

2 Grundlagen

Transport Layer Security (TLS), früher bekannt als Secure Socket Layer (SSL), ermöglicht die sichere Übertragung von Informationen aus der Anwendungsschicht (zum Beispiel HTTPS, FTPS oder IMAPS) über TCP/IP-basierte Verbindungen (insbesondere das Internet).

Bevor Daten übertragen werden können, muss zunächst eine gesicherte Verbindung zwischen den beiden Verbindungspartnern (Client und Server) aufgebaut werden. Dieser Vorgang heißt *Handshake* und ist ein wichtiger Bestandteil des TLS-Protokolls. Hierbei werden zwischen Client und Server vereinbart:

1. Kryptographische Verfahren für die *Verschlüsselung*, *Integritätssicherung*, *Schlüsseleinigung* und ggf. für die (ein- oder beidseitige) *Authentisierung*. Diese Verfahren werden durch sogenannte Cipher-Suiten und weitere kryptographische Parameter festgelegt (siehe Abschnitte 3.3 und 3.4).
2. Ein gemeinsames Geheimnis, das *master secret*. Aus diesem werden von beiden Verbindungspartnern die Sitzungsschlüssel für den Integritätsschutz und die Verschlüsselung abgeleitet.

Hinweis: Das TLS-Protokoll erlaubt auch Verbindungen, die nicht oder nur einseitig authentisiert sind (beispielsweise sind HTTPS-Verbindungen üblicherweise nur serverseitig authentisiert). Daher sollten Entwickler kryptographischer Systeme darauf achten, ob eine weitere Authentisierung innerhalb der Anwendungsschicht erforderlich ist (Beispiel: Authentisierung eines Homebanking-Benutzers durch Anforderung eines Passwortes). Bei besonders kritischen Operationen sollte dabei grundsätzlich eine Authentisierung durch Wissen und Besitz (Zwei-Faktor-Authentisierung) erfolgen, die sich unter Ausnutzung kryptographischer Mechanismen auch auf die übertragenen Daten erstrecken sollte.

3 Empfehlungen

3.1 Allgemeine Hinweise

3.1.1 Verwendungszeiträume

Die Empfehlungen in dieser Technischen Richtlinie sind mit Verwendungszeiträumen versehen. Die Angabe der Jahreszahl bedeutet hierbei, dass das entsprechende Verfahren bis zum Ende des angegebenen Jahres empfohlen wird. Ist die Jahreszahl mit einem „+“-Zeichen gekennzeichnet, so bedeutet dies, dass dieser Verwendungszeitraum möglicherweise in einer zukünftigen Version dieser Technischen Richtlinie verlängert wird.

3.1.2 Sicherheitsniveau

Das Sicherheitsniveau für alle kryptographischen Verfahren in dieser Technischen Richtlinie richtet sich nach dem in Abschnitt 1.1 in [TR-02102-1] angegebenen Sicherheitsniveau. Es liegt zurzeit bei 100 Bit.

Hinweis: Ab dem Jahr 2023 wird ein Sicherheitsniveau von 120 Bit angestrebt. Als Übergangsregelung ist die Verwendung von RSA-basierten Signatur- und Verschlüsselungsverfahren mit einer Schlüssellänge ab 2000 Bit für das gesamte Jahr 2023 aber weiter konform zu dieser Richtlinie. Siehe dazu auch Abschnitt 1.1 in [TR-02102-1].

3.1.3 Schlüssellängen bei Verfahren mit elliptischen Kurven

Für einen Einsatzzeitraum bis Ende 2022 ist das Sicherheitsniveau bei Verfahren, die auf elliptischen Kurven basieren, etwas größer (im Vergleich zu RSA) gewählt worden, um einen Sicherheitsspielraum für diese Verfahren zu erreichen (vgl. Abschnitt 3.6). Für eine Begründung und weitere Erläuterungen siehe Bemerkung 4, Kapitel 3 in [TR-02102-1].

3.2 SSL/TLS-Versionen

Das SSL-Protokoll existiert in den Versionen 1.0, 2.0 und 3.0, wobei die Version 1.0 nicht veröffentlicht wurde. TLS 1.0 ist eine direkte Weiterentwicklung von SSL 3.0 und wird in [RFC2246] spezifiziert. Des Weiteren gibt es die TLS-Versionen 1.1, 1.2 und 1.3, welche in [RFC4346], [RFC5246] und [RFC8446] spezifiziert werden.

Empfehlungen für die Wahl der TLS-Version sind:

- Grundsätzlich werden TLS 1.2 und TLS 1.3 empfohlen.
- TLS 1.0 und TLS 1.1 werden **nicht empfohlen** (siehe auch Abschnitt 3.3.1.4).
- SSL v2 ([SSLv2]) und SSL v3 ([SSLv3]) werden **nicht empfohlen** (siehe auch [RFC6176] und [RFC7568]).

3.3 Empfehlungen zu TLS 1.2

In TLS 1.2 werden die kryptographischen Verfahren einer Verbindung durch eine Cipher-Suite festgelegt. Eine Cipher-Suite spezifiziert ein (authentisiertes) Schlüsseleinigungsverfahren für das Handshake-Protokoll, ein authentisiertes Verschlüsselungsverfahren für das Record-Protokoll und eine Hashfunktion

für die Schlüsselableitung. Für die Schlüsseleinigung müssen je nach Cipher-Suite noch eine Diffie-Hellman Gruppe (in einem endlichen Körper oder über einer elliptischen Kurve) oder Signaturverfahren festgelegt werden.

Eine vollständige Liste aller definierten Cipher-Suiten mit Verweisen auf die jeweiligen Spezifikationen ist verfügbar unter [IANA].

3.3.1 Cipher-Suiten

In TLS 1.2 werden Cipher-Suiten in der Regel mit der Namenskonvention `TLS_AKE_WITH_Enc_Hash` angegeben, wobei *AKE* ein (authentisiertes) Schlüsseleinigungsverfahren, *Enc* ein Verschlüsselungsverfahren mit Betriebsmodus und *Hash* eine Hashfunktion bezeichnet. Die Funktion *Hash* wird von einem HMAC (*Keyed-Hash Message Authentication Code*) genutzt, der für die PRF (*Pseudo-Random Function*) zur Schlüsselableitung verwendet wird.[1] Falls *Enc* kein authentisiertes Verschlüsselungsverfahren (*Authenticated Encryption with Associated Data*, kurz AEAD) ist, so wird der HMAC zusätzlich für die Integritätssicherung im Record-Protokoll eingesetzt.

Grundsätzlich wird empfohlen, nur Cipher-Suiten einzusetzen, die die Anforderungen an die Algorithmen und Schlüssellängen der [TR-02102-1] erfüllen.

3.3.1.1 (EC)DHE Cipher-Suiten

Die folgenden Cipher-Suiten mit Perfect Forward Secrecy[2] werden empfohlen:

Cipher-Suite	IANA-Nr.	Referenziert in	Verwendung bis
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	0xC0,0x23	[RFC5289]	2026+
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	0xC0,0x24	[RFC5289]	2026+
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	0xC0,0x2B	[RFC5289]	2026+
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	0xC0,0x2C	[RFC5289]	2026+
TLS_ECDHE_ECDSA_WITH_AES_128_CCM	0xC0,0xAC	[RFC7251]	2026+
TLS_ECDHE_ECDSA_WITH_AES_256_CCM	0xC0,0xAD	[RFC7251]	2026+
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	0xC0,0x27	[RFC5289]	2026+
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	0xC0,0x28	[RFC5289]	2026+
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	0xC0,0x2F	[RFC5289]	2026+
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	0xC0,0x30	[RFC5289]	2026+
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	0x00,0x40	[RFC5246]	2026+
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	0x00,0x6A	[RFC5246]	2026+
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	0x00,0xA2	[RFC5288]	2026+
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	0x00,0xA3	[RFC5288]	2026+

1 Bei Cipher-Suiten, die den Betriebsmodus CCM verwenden, ist keine Hashfunktion angegeben. Diese Cipher-Suiten verwenden SHA-256 für die PRF.

2 Perfect Forward Secrecy (kurz PFS, auch Forward Secrecy) bedeutet, dass eine Verbindung auch bei Kenntnis der Langzeit-Schlüssel der Kommunikationspartner nicht nachträglich entschlüsselt werden kann. Bei der Verwendung von TLS zum Schutz personenbezogener oder anderer sensibler Daten wird Perfect Forward Secrecy grundsätzlich empfohlen.

Cipher-Suite	IANA-Nr.	Referenziert in	Verwendung bis
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	0x00,0x67	[RFC5246]	2026+
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	0x00,0x6B	[RFC5246]	2026+
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	0x00,0x9E	[RFC5288]	2026+
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	0x00,0x9F	[RFC5288]	2026+
TLS_DHE_RSA_WITH_AES_128_CCM	0xC0,0x9E	[RFC6655]	2026+
TLS_DHE_RSA_WITH_AES_256_CCM	0xC0,0x9F	[RFC6655]	2026+

Tabelle 1: Empfohlene Cipher-Suiten für TLS 1.2 mit Perfect Forward Secrecy

3.3.1.2 (EC)DH Cipher Suiten

Sofern die Verwendung der in Abschnitt 3.3.1.1 empfohlenen Cipher-Suiten mit Perfect Forward Secrecy nicht möglich ist, können auch die folgenden Cipher-Suiten eingesetzt werden:

Cipher-Suite	IANA-Nr.	Referenziert in	Verwendung bis
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	0xC0,0x25	[RFC5289]	2026+
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	0xC0,0x26	[RFC5289]	2026+
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	0xC0,0x2D	[RFC5289]	2026+
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	0xC0,0x2E	[RFC5289]	2026+
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	0xC0,0x29	[RFC5289]	2026+
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	0xC0,0x2A	[RFC5289]	2026+
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	0xC0,0x31	[RFC5289]	2026+
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	0xC0,0x32	[RFC5289]	2026+
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	0x00,0x3E	[RFC5246]	2026+
TLS_DH_DSS_WITH_AES_256_CBC_SHA256	0x00,0x68	[RFC5246]	2026+
TLS_DH_DSS_WITH_AES_128_GCM_SHA256	0x00,0xA4	[RFC5288]	2026+
TLS_DH_DSS_WITH_AES_256_GCM_SHA384	0x00,0xA5	[RFC5288]	2026+
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	0x00,0x3F	[RFC5246]	2026+
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	0x00,0x69	[RFC5246]	2026+
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	0x00,0xA0	[RFC5288]	2026+
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	0x00,0xA1	[RFC5288]	2026+

Tabelle 2: Empfohlene Cipher-Suiten für TLS 1.2 ohne Perfect Forward Secrecy

3.3.1.3 Schlüsseleinigung mit vorab ausgetauschten Daten

Sollen bei einer TLS-Verbindung zusätzliche vorab ausgetauschte Daten in die Schlüsseleinigung einfließen, können Cipher-Suiten mit einem Pre-shared Key (kurz PSK) verwendet werden. Grundsätzlich werden

hierbei solche Cipher-Suiten empfohlen, bei denen neben dem Pre-shared Key weitere ephemere Schlüssel oder ausgetauschte Zufallszahlen in die Schlüsseleinigung eingehen.

Die Verwendung von Cipher-Suiten vom Typ `TLS_PSK_*`, das heißt ohne zusätzliche ephemere Schlüssel oder Zufallszahlen, wird **nicht empfohlen**, da bei diesen Cipher-Suiten die Sicherheit der Verbindung ausschließlich auf der Entropie und der Vertraulichkeit des Pre-shared Keys beruht.

Die folgenden Cipher-Suiten mit PSK werden empfohlen:

Cipher-Suite	IANA-Nr.	Referenziert in	Verwendung bis
TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256	0xC0,0x37	[RFC5489]	2026+
TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384	0xC0,0x38	[RFC5489]	2026+
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	0xD0,0x01	[RFC8442]	2026+
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	0xD0,0x02	[RFC8442]	2026+
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	0xD0,0x05	[RFC8442]	2026+
TLS_DHE_PSK_WITH_AES_128_CBC_SHA256	0x00,0xB2	[RFC5487]	2026+
TLS_DHE_PSK_WITH_AES_256_CBC_SHA384	0x00,0xB3	[RFC5487]	2026+
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	0x00,0xAA	[RFC5487]	2026+
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	0x00,0xAB	[RFC5487]	2026+
TLS_DHE_PSK_WITH_AES_128_CCM	0xC0,0xA6	[RFC6655]	2026+
TLS_DHE_PSK_WITH_AES_256_CCM	0xC0,0xA7	[RFC6655]	2026+
TLS_RSA_PSK_WITH_AES_128_CBC_SHA256	0x00,0xB6	[RFC5487]	2026+
TLS_RSA_PSK_WITH_AES_256_CBC_SHA384	0x00,0xB7	[RFC5487]	2026+
TLS_RSA_PSK_WITH_AES_128_GCM_SHA256	0x00,0xAC	[RFC5487]	2026+
TLS_RSA_PSK_WITH_AES_256_GCM_SHA384	0x00,0xAD	[RFC5487]	2026+

Tabelle 3: Empfohlene Cipher-Suiten für TLS 1.2 mit Pre-Shared Key

Hinweis: Die Cipher-Suiten der Form `TLS_RSA_PSK_*` aus Tabelle 3 bieten keine Perfect Forward Secrecy, alle anderen Cipher-Suiten aus Tabelle 3 hingegen bieten Perfect Forward Secrecy.

3.3.1.4 Übergangsregelungen

SHA-1 ist keine kollisionsresistente Hashfunktion; die Erzeugung von SHA-1-Kollisionen ist zwar mit einigem Aufwand verbunden, aber praktisch machbar [SBK17, LP19]. Gegen die Verwendung in Konstruktionen, die keine Kollisionsresistenz benötigen (zum Beispiel als Grundlage für einen HMAC oder als Komponente eines Pseudozufallsgenerators) spricht aber nach gegenwärtigem Kenntnisstand sicherheitstechnisch nichts. Es wird empfohlen, auch in diesen Anwendungen als grundsätzliche Sicherungsmaßnahme eine Hashfunktion der SHA-2-Familie oder der SHA-3-Familie einzusetzen. Prinzipiell ist eine Verwendung von SHA-1 in der HMAC-Konstruktion oder in anderen kryptographischen Mechanismen mit vergleichbaren kryptographischen Anforderungen an die genutzte Hashfunktion (zum Beispiel im Rahmen eines Pseudozufallsgenerators oder als Teil der Mask Generation Function in RSA-OAEP) bis 2019 konform zu der vorliegenden Technischen Richtlinie.

Daher kann, abweichend zu den Empfehlungen in diesem Kapitel und in Teil 1 dieser Technischen Richtlinie [TR-02102-1], in *bestehenden* TLS-Anwendungen als Hashfunktion für die Integritätssicherung mittels HMAC übergangsweise noch SHA-1 eingesetzt werden (das heißt Cipher-Suiten der Form `*_SHA`).

Unabhängig von dem in Tabelle 4 angegebenen *maximalen* Verwendungszeitraum wird eine schnellstmögliche Migration zu SHA-256 bzw. SHA-384 und TLS 1.2 empfohlen.

Hinweis: Da TLS 1.1 die Hashfunktion SHA-1 als Komponente für die Signaturerstellung verwendet (und keine Unterstützung der SHA-2-Familie bietet), wird der Einsatz von TLS 1.1 nicht mehr empfohlen.

Der Verschlüsselungsalgorithmus RC4 in TLS weist erhebliche Sicherheitsschwächen auf. Seine Verwendung wird daher nicht empfohlen (siehe auch [RFC7465]).

Abweichung	Verwendung maximal bis	Empfehlung
SHA-1 zur HMAC-Berechnung und als Komponente der PRF in TLS	2019	Migration zu SHA-256/-384
SHA-1 als Komponente für die Signaturerstellung in TLS	2015	Migration zu SHA-256/-384/-512

Tabelle 4: Übergangsregelungen für TLS 1.2 und frühere TLS-Versionen

Anmerkung: In der vorliegenden Technischen Richtlinie wurde der Einsatz von SHA-1 bis 2015 als Komponente für die Signaturerstellung *ausschließlich im Rahmen von TLS* empfohlen (vgl. Tabelle 4), damit TLS 1.0 noch übergangsweise bis Ende 2015 eingesetzt werden konnte. Da SHA-1 für den Handshake bei TLS 1.0 erforderlich ist und eine SHA-1-Kollision sicherlich nicht in Echtzeit (also während des Handshakes) berechnet werden konnte, war der Einsatz von SHA-1 in diesem einzigen Spezialfall noch etwas länger möglich.

Grundsätzlich wird jedoch die Verwendung von SHA-1 (z.B. für die Erstellung von Signaturen) in der TR-02102-1 seit 2013 nicht mehr empfohlen.

3.3.2 Diffie-Hellman Gruppen

Bei Verwendung von `TLS_DHE_*` oder `TLS_ECDHE_*` Cipher-Suiten kann der Client dem Server mittels der „supported_groups“ Erweiterung (ehemals auch „elliptic_curves“ Erweiterung genannt) signalisieren, welche Diffie-Hellman Gruppen er verwenden möchte (siehe [RFC7919] für DHE und [RFC8422] für ECDHE).

Die Verwendung der „supported_groups“ Erweiterung für `TLS_ECDHE_*` Cipher-Suiten wird empfohlen.

Die Verwendung der „supported_groups“ Erweiterung für `TLS_DHE_*` Cipher-Suiten wird empfohlen, sobald geeignete Implementierungen zur Verfügung stehen.

Die folgenden Diffie-Hellman Gruppen werden empfohlen:

Diffie-Hellman Gruppe	IANA-Nr.	Referenziert in	Verwendung bis
`secp256r1`	23	[RFC8422]	2026+
`secp384r1`	24	[RFC8422]	2026+
`brainpoolP256r1`	26	[RFC7027]	2026+
`brainpoolP384r1`	27	[RFC7027]	2026+
`brainpoolP512r1`	28	[RFC7027]	2026+
`ffdhe2048`	256	[RFC7919]	2022
`ffdhe3072`	257	[RFC7919]	2026+
`ffdhe4096`	258	[RFC7919]	2026+

Tabelle 5: Empfohlene Diffie-Hellman Gruppen für TLS 1.2

Grundsätzlich ist Abschnitt 3.6 bei der Wahl von Domainparametern und Schlüssellängen zu beachten.

3.3.3 Signaturverfahren

In TLS 1.2 kann der Client dem Server mittels der „signature_algorithms“ Erweiterung (siehe [RFC5246]) signalisieren, welche Signaturverfahren er für die Schlüsseleinigung und für Zertifikate akzeptiert. Der Algorithmus muss dabei als Kombination aus Signaturverfahren und Hashfunktion angegeben werden.

Die Verwendung der „signature_algorithms“ Erweiterung wird empfohlen.

Die folgenden Signaturverfahren werden empfohlen:

Signaturverfahren	IANA-Nr.	Referenziert in	Verwendung bis
`rsa`	1	[RFC5246]	2025
`dsa`	2	[RFC5246]	2026+
`ecdsa`	3	[RFC5246]	2026+

Tabelle 6: Empfohlene Signaturverfahren für TLS 1.2

Für Domainparameter und Schlüssellängen ist Abschnitt 3.6 zu beachten.

Hinweis: Die Nutzung des Signaturverfahrens `rsa` (IANA-Nr. 1) ist auf Grund der Verwendung des PKCS #1 v1.5 Paddingverfahrens nur noch bis 2025 empfohlen (siehe auch Abschnitt 1.4 in [TR-02102-1]).

Die folgenden Hashfunktionen werden (in Kombination mit einem Signaturverfahren aus Tabelle 6) empfohlen:

Hashfunktion	IANA-Nr.	Referenziert in	Verwendung bis
`sha256`	4	[RFC5246]	2026+
`sha384`	5	[RFC5246]	2026+
`sha512`	6	[RFC5246]	2026+

Tabelle 7: Empfohlene Hashfunktionen für Signaturverfahren in TLS 1.2

3.3.4 Weitere Empfehlungen

3.3.4.1 Session Renegotiation

Es wird empfohlen Session Renegotiation nur auf Basis von [RFC5746] zu verwenden. Durch den Client initiierte Renegotiation sollte vom Server abgelehnt werden.

3.3.4.2 Verkürzung der HMAC-Ausgabe

Die in [RFC6066] definierte Extension „truncated_hmac“ zur Verkürzung der Ausgabe des HMAC auf 80 Bit sollte *nicht* verwendet werden.

3.3.4.3 TLS-Kompression und der CRIME-Angriff

TLS bietet die Möglichkeit, die übertragenen Daten vor der Verschlüsselung zu komprimieren. Dies führt zu der Möglichkeit eines Seitenkanalangriffes auf die Verschlüsselung, und zwar mit Hilfe der Länge der verschlüsselten Daten (siehe [CRIME]).

Um dies zu verhindern, muss sichergestellt werden, dass alle Daten eines Datenpakets von dem korrekten und legitimen Verbindungspartner stammen und keine Plaintext-Injection durch einen Angreifer möglich ist. Kann dies nicht sichergestellt werden, so wird empfohlen die TLS-Datenkompression nicht zu verwenden.

3.3.4.4 Der Lucky 13 Angriff

Lucky 13 ist ein Seitenkanalangriff (Timing) gegen Cipher-Suiten mit CBC-Modus, bei dem der Angreifer sehr geringe Zeitdifferenzen bei der Verarbeitung des Paddings auf Seiten des Servers ausnutzt. Für diesen Angriff muss der Angreifer sehr genaue Zeitmessungen im Netzwerk machen können. Er schickt manipulierte Chiffrate an den Server und misst die Zeit, die der Server benötigt, um das Padding dieser Chiffrate zu prüfen bzw. einen Fehler zu melden. Durch Netzwerk-Jitter können hier aber sehr leicht Fehler bei der Zeitmessung entstehen, so dass ein Angriff grundsätzlich als schwierig realisierbar erscheint, denn der Angreifer muss im Netzwerk „sehr nahe“ am Server sein, um genau genug messen zu können.

Der Angriff kann abgewehrt werden, wenn

- Authenticated Encryption, wie zum Beispiel AES-GCM oder AES-CCM, oder
- Encrypt-then-MAC (siehe auch nächster Abschnitt)

eingesetzt wird.

3.3.4.5 Die Encrypt-then-MAC Erweiterung

Gemäß TLS-Spezifikation (siehe [RFC5246]) werden die zu übertragenen Daten zunächst mit einem Message Authentication Code (MAC) gesichert und dann mit einem Padding versehen; danach werden die Daten und das Padding verschlüsselt. Diese Reihenfolge („MAC-then-Encrypt“) war in der Vergangenheit häufig der Grund für Angriffe auf die Verschlüsselung, da das Padding nicht durch den MAC geschützt ist.

Bei den sogenannten Padding-Oracle-Angriffen werden die verschlüsselten TLS-Pakete durch einen Man-in-the-Middle-Angreifer manipuliert, um die Prüfung des Paddings als Seitenkanal zu missbrauchen. Dies kann beispielsweise dazu führen, dass der Angreifer ein HTTPS-Sitzungs-Cookie entschlüsseln kann und somit die Sitzung des Opfers übernehmen kann.

In [RFC7366] wird die TLS-Erweiterung „Encrypt-then-MAC“ spezifiziert. Hierbei werden die zu übertragenen Daten zuerst mit einem Padding versehen, dann verschlüsselt und danach mit einem MAC gesichert. Damit sind Manipulationen des Paddings ausgeschlossen, da es auch durch den MAC gesichert ist.

Der Einsatz der TLS-Erweiterung „Encrypt-then-MAC“ gemäß [RFC7366] wird empfohlen, sobald geeignete Implementierungen zur Verfügung stehen.

3.3.4.6 Die Heartbeat Erweiterung

Die Heartbeat-Erweiterung wird in [RFC6520] spezifiziert; sie ermöglicht es, eine TLS-Verbindung über einen längeren Zeitraum aufrecht zu halten, ohne eine Renegotiation der Verbindung durchführen zu müssen. Durch den sogenannten Heartbleed-Bug ist es einem Angreifer möglich, bestimmte Speicherbereiche des Servers auszulesen, die möglicherweise geheimes Schlüsselmaterial enthalten. Dies kann zu einer vollständigen Kompromittierung des Servers führen, falls der private Schlüssel des Servers bekannt wird.

Empfehlung: Es wird dringend empfohlen, die Heartbeat-Erweiterung nicht zu verwenden. Sollte es trotzdem erforderlich sein, so sollte sichergestellt sein, dass die verwendete TLS-Implementierung nicht anfällig für den Heartbleed-Bug ist.

3.3.4.7 Die Extended Master Secret Erweiterung

Um Angriffe, wie zum Beispiel den Triple Handshake-Angriff (siehe [BDF14]) abzuwehren, ist es sehr sinnvoll, weitere Verbindungsparameter in den TLS-Handshake einfließen zu lassen, damit unterschiedliche TLS-Verbindungen auch unterschiedliche Master Secrets (aus welchem die symmetrischen Schlüssel abgeleitet werden) benutzen.

In [RFC7627] wird die TLS-Erweiterung *Extended Master Secret* spezifiziert, die bei der Berechnung des „erweiterten" Master Secrets einen Hashwert über alle Nachrichten des TLS-Handshakes mit in dieses einfließen lässt.

Der Einsatz der TLS-Erweiterung *Extended Master Secret* gemäß [RFC7627] wird empfohlen, sobald geeignete Implementierungen zur Verfügung stehen.

3.4 Empfehlungen zu TLS 1.3

In TLS 1.3 werden die kryptographischen Verfahren einer Verbindung durch einen Handshake Modus, eine Diffie-Hellman Gruppe (bei (EC)DHE), ein Signaturverfahren (bei zertifikatsbasierter Authentisierung) und eine Cipher-Suite festgelegt. Im Gegensatz zu früheren TLS-Versionen spezifiziert eine Cipher-Suite hierbei nur ein authentisiertes Verschlüsselungsverfahren für das Record-Protokoll sowie eine Hashfunktion für die Schlüsselableitung.

3.4.1 Handshake Modi

Neben der standardmäßigen Diffie-Hellman Schlüsseleinigung über endlichen Körpern (DHE) oder elliptischen Kurven (ECDHE) gibt es in TLS 1.3 weitere Handshake Modi, die Pre-shared Keys (PSK) verwenden. Unter Pre-shared Keys versteht man hierbei Schlüsselmaterial, das entweder vorab verteilt wurde oder das in einer vergangenen Session über den Session-Ticket-Mechanismus ausgetauscht wurde.

Die folgenden PSK-Modi werden empfohlen:

PSK-Modus	IANA-Nr.	Referenziert in	Verwendung bis
`psk_ke`	0	[RFC8446]	2026+
`psk_dhe_ke`	1	[RFC8446]	2026+

Tabelle 8: Empfohlene Pre-Shared Key Modi für TLS 1.3

Hinweis: Der PSK-Modus `psk_ke` bietet keine Perfect Forward Secrecy. Dieser Modus sollte daher nur in speziellen Anwendungsfällen nach Hinzuziehen eines Experten eingesetzt werden.

Hinweis: Bei PSK-Handshakes besteht die Möglichkeit, Anwendungsdaten bereits mit der ersten Handshake-Nachricht mitzuschicken (*zero round-trip time* Daten, kurz 0-RTT Daten). Diese Daten sind nicht gegen Replay-Angriffe geschützt. Das Senden oder Annehmen von 0-RTT Daten wird daher **nicht empfohlen**.

3.4.2 Diffie-Hellman Gruppen

In TLS 1.3 können die Kommunikationspartner mittels der „supported_groups" Erweiterung signalisieren, welche Diffie-Hellman Gruppen für (EC)DHE verwendet werden sollen.

Die folgenden Diffie-Hellman Gruppen werden empfohlen:

Diffie-Hellman Gruppe	IANA-Nr.	Referenziert in	Verwendung bis
secp256r1	23	[RFC8422]	2026+
secp384r1	24	[RFC8422]	2026+
brainpoolP256r1tls13	31	[RFC8734]	2026+
brainpoolP384r1tls13	32	[RFC8734]	2026+
brainpoolP512r1tls13	33	[RFC8734]	2026+
ffdhe2048	256	[RFC7919]	2022
ffdhe3072	257	[RFC7919]	2026+
ffdhe4096	258	[RFC7919]	2026+

Tabelle 9: Empfohlene Diffie-Hellman Gruppen für TLS 1.3

Hinweis: Die Brainpool-Kurven werden grundsätzlich empfohlen.

Hinweis: In [RFC8446] wurden die IANA-Nummern einiger EC-Gruppen, die laut [RFC8446] entweder veraltet sind oder wenig genutzt wurden, als „obsolete_RESERVED“ gekennzeichnet. Dazu zählen auch die IANA-Nummern 26, 27, 28, die für die Brainpool-Kurven zur Nutzung in TLS 1.2 und früheren TLS-Versionen registriert sind. Aus diesem Grund wurden für die Nutzung der Brainpool-Kurven in TLS 1.3 die IANA-Nummern 31, 32, 33 reserviert (siehe [RFC8734]).

3.4.3 Signaturverfahren

In TLS 1.3 können die Kommunikationspartner mittels der Erweiterungen „signature_algorithms“ und „signature_algorithms_cert“ signalisieren, welche Signaturverfahren zur zertifikatsbasierten Authentisierung verwenden werden sollen. Die „signature_algorithms“ Erweiterung bezieht sich dabei auf Signaturen, die der Client oder Server für eine CertificateVerify-Nachricht erstellt, und die „signature_algorithms_cert“ Erweiterung auf Zertifikatssignaturen.

Die folgenden Signaturverfahren werden für die „signature_algorithms“ Erweiterung empfohlen:

Signaturverfahren	IANA-Nr.	Referenziert in	Verwendung bis
rsa_pss_rsae_sha256	0x0804	[RFC8446]	2026+
rsa_pss_rsae_sha384	0x0805	[RFC8446]	2026+
rsa_pss_rsae_sha512	0x0806	[RFC8446]	2026+
rsa_pss_pss_sha256	0x0809	[RFC8446]	2026+
rsa_pss_pss_sha384	0x080A	[RFC8446]	2026+
rsa_pss_pss_sha512	0x080B	[RFC8446]	2026+
ecdsa_secp256r1_sha256	0x0403	[RFC8446]	2026+
ecdsa_secp384r1_sha384	0x0503	[RFC8446]	2026+
ecdsa_brainpoolP256r1tls13_sha256	0x081A	[RFC8734]	2026+
ecdsa_brainpoolP384r1tls13_sha384	0x081B	[RFC8734]	2026+
ecdsa_brainpoolP512r1tls13_sha512	0x081C	[RFC8734]	2026+

Tabelle 10: Empfohlene Signaturverfahren für TLS 1.3 (Client-/Server-Signatur)

Die folgenden Algorithmen werden für die „signature_algorithms_cert“ Erweiterung empfohlen:

Signaturverfahren	IANA-Nr.	Referenziert in	Verwendung bis
rsa_pkcs1_sha256	0x0401	[RFC8446]	2025
rsa_pkcs1_sha384	0x0501	[RFC8446]	2025
rsa_pkcs1_sha512	0x0601	[RFC8446]	2025
rsa_pss_rsae_sha256	0x0804	[RFC8446]	2026+
rsa_pss_rsae_sha384	0x0805	[RFC8446]	2026+
rsa_pss_rsae_sha512	0x0806	[RFC8446]	2026+
rsa_pss_pss_sha256	0x0809	[RFC8446]	2026+
rsa_pss_pss_sha384	0x080A	[RFC8446]	2026+
rsa_pss_pss_sha512	0x080B	[RFC8446]	2026+
ecdsa_secp256r1_sha256	0x0403	[RFC8446]	2026+
ecdsa_secp384r1_sha384	0x0503	[RFC8446]	2026+
ecdsa_brainpoolP256r1tls13_sha256	0x081A	[RFC8734]	2026+
ecdsa_brainpoolP384r1tls13_sha384	0x081B	[RFC8734]	2026+
ecdsa_brainpoolP512r1tls13_sha512	0x081C	[RFC8734]	2026+

Tabelle 11: Empfohlene Signaturverfahren für TLS 1.3 (Zertifikatssignaturen)

Für Schlüssellängen bei RSA-Signaturen ist Abschnitt 3.6 zu beachten.

Hinweis: Die Nutzung der Signaturverfahren `rsa_pkcs1_*` (IANA-Nr. 0x0401, 0x0501 und 0x0601) ist auf Grund der Verwendung des PKCS #1 v1.5 Paddingverfahrens nur noch bis 2025 empfohlen (siehe auch Abschnitt 1.4 in [TR-02102-1]).

3.4.4 Cipher-Suiten

In TLS 1.3 werden Cipher-Suiten mit der Namenskonvention `TLS_`*`AEAD`*`_`*`Hash`* angegeben, wobei *`AEAD`* ein authentisiertes Verschlüsselungsverfahren (*authenticated encryption with associated data*, kurz AEAD) für das Record-Protokoll und *`Hash`* eine Hashfunktion für die Nutzung mit HMAC (*Keyed-Hash Message Authentication Code*) und HKDF (*HMAC-based Extract-and-Expand Key Derivation Function*) im Handshake-Protokoll bezeichnet.

Die folgenden Cipher-Suiten werden empfohlen:

Cipher-Suite	IANA-Nr.	Referenziert in	Verwendung bis
TLS_AES_128_GCM_SHA256	0x13,0x01	[RFC8446]	2026+
TLS_AES_256_GCM_SHA384	0x13,0x02	[RFC8446]	2026+
TLS_AES_128_CCM_SHA256	0x13,0x04	[RFC8446]	2026+

Tabelle 12: Empfohlene Cipher-Suiten für TLS 1.3

3.5 Authentisierung der Kommunikationspartner

Das TLS-Protokoll bietet die folgenden drei Möglichkeiten zur Authentisierung der Kommunikationspartner:

- Authentisierung beider Kommunikationspartner
- Nur serverseitige Authentisierung
- Keine Authentisierung

Die Notwendigkeit einer Authentisierung ist abhängig von der jeweiligen Anwendung. Bei der Verwendung von TLS im Web ist im Allgemeinen zumindest eine Authentisierung des Servers notwendig. Bei der Verwendung in geschlossenen Systemen (VPN o. ä.) ist zumeist eine beidseitige Authentisierung notwendig.

Für die Authentisierung innerhalb von Projekten des Bundes sind die Vorgaben der Technischen Richtlinie [TR-03116-4] in der jeweils aktuellen Fassung zu beachten.

3.6 Domainparameter und Schlüssellängen

Die Domainparameter und Schlüssellängen für

- statische Schlüsselpaare der Kommunikationspartner,
- ephemere Schlüsselpaare bei der Verwendung von Cipher-Suiten mit Perfect Forward Secrecy, und
- Schlüsselpaare für die Signatur von Zertifikaten

müssen den Vorgaben in Teil 1 dieser Technischen Richtlinie (siehe [TR-02102-1]) entsprechen.

3.6.1 Schlüssellängen

Es wird empfohlen, mindestens die folgenden Schlüssellängen zu verwenden:

Algorithmus	Minimale Schlüssellänge	Verwendung spätestens ab	Verwendung bis
Signaturschlüssel für Zertifikate und Schlüsseleinigung			
ECDSA	224 Bit		2015
	250 Bit		2026+
DSS	2000 Bit		2022
	3000 Bit	2023	2026+
RSA	2000 Bit		2023
	3000 Bit	2024	2026+
Statische und ephemere Diffie-Hellman-Schlüssel			
ECDH	224 Bit		2015
	250 Bit		2026+
DH	2000 Bit		2022
	3000 Bit	2023	2026+

Tabelle 13: Empfohlene Mindest-Schlüssellängen für das TLS-Handshakeprotokoll

Hinweis: Ist ein Schlüsselpaar *statisch*, so wird es mehrfach für neue Verbindungen wiederverwendet. Im Gegensatz dazu bedeutet *ephemer*, dass für jede neue Verbindung auch ein neues Schlüsselpaar erzeugt und verwendet wird. Ephemere Schlüssel müssen nach Verbindungsende unbedingt sicher gelöscht werden, siehe dazu auch Abschnitt 4.2. Soll eine Verbindung die Eigenschaft *Perfect Forward Secrecy* erfüllen, müssen ausschließlich ephemere Schlüsselpaare verwendet werden.

Wichtiger Hinweis: Es ist sinnvoll, für RSA, DH und DSS eine Schlüssellänge von 3000 Bit zu nutzen, um ein gleichartiges Sicherheitsniveau für alle asymmetrischen Verfahren zu erreichen. Eine Schlüssellänge von mindestens 3000 Bit ist damit ab dem Jahr 2023 für kryptographische Implementierungen verbindlich, wenn sie zur vorliegenden Technischen Richtlinie konform sein sollen. Jede Schlüssellänge von mindestens 2000 Bit bleibt aber für Systeme mit einer Lebensdauer bis zum Jahr 2022 konform zur vorliegenden Technischen Richtlinie. Als Übergangsregelung ist außerdem die Nutzung von RSA-Schlüsseln mit einer Länge ab 2000 Bit bis Ende 2023 ebenfalls noch konform. Es handelt sich dabei um die empfohlene Mindest-Schlüssellänge für RSA, DH und DSS. Weitere Informationen finden sich in den Bemerkungen 4 und 5 in Kapitel 3 in [TR-02102-1].

Bemerkung: Die Empfehlungen in dieser Technischen Richtlinie sind geeignet, um das in Abschnitt 3.1.2 genannte Sicherheitsniveau von zurzeit 100 Bit zu erreichen.

Der Vorhersagezeitraum für die vorliegenden Empfehlungen beträgt 7 Jahre. Geeignete Empfehlungen für deutlich größere Zeiträume, wie sie in anderen öffentlich verfügbaren Dokumenten zu finden sind, sind naturgemäß sehr schwierig, da zukünftige kryptographische Entwicklungen über längere Zeiträume nicht oder zumindest nicht präzise vorausgesagt werden können. In solchen Fällen umfassen diese Empfehlungen Parameter und Schlüssellängen, die über die in der vorliegenden Technischen Richtlinie hinausgehen können.

3.6.2 Verwendung von elliptischen Kurven

Bei der Verwendung von elliptischen Kurven werden stets kryptographisch starke Kurven über endlichen Körpern der Form F_p (p prim) empfohlen. Zusätzlich wird empfohlen, nur *named curves* (siehe Abschnitt „Supported Groups Registry“ in [IANA]) einzusetzen, um Angriffe über nicht verifizierte schwache Domainparameter zu verhindern. Die folgenden *named curves* werden empfohlen:

- brainpoolP256r1, brainpoolP384r1, brainpoolP512r1 (siehe [RFC5639] und [RFC7027])

Sollten diese Kurven nicht verfügbar sein, so können auch die folgenden Kurven eingesetzt werden:

- secp256r1, secp384r1

4 Schlüssel und Zufallszahlen

4.1 Schlüsselspeicherung

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 zum Beispiel durch die Verwendung zertifizierter Hardware (Chipkarte, HSM) gewährleistet werden.

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

4.2 Umgang mit Ephemer-Schlüsseln

Wenn eine Cipher-Suite mit Perfect Forward Secrecy verwendet wird, sollte sichergestellt werden, dass alle Ephemer-Schlüssel nach ihrer Verwendung unwiderruflich gelöscht werden, und keine Kopien dieser Schlüssel erzeugt wurden. Ephemer- bzw. Sitzungsschlüssel sollten nur für *eine* Verbindung benutzt werden und grundsätzlich nicht persistent abgespeichert werden.

4.3 Zufallszahlen

Für die Erzeugung von Zufallszahlen, zum Beispiel für kryptographische Schlüssel oder die Signaturerzeugung, müssen geeignete Zufallszahlengeneratoren eingesetzt werden.

Empfohlen wird ein Zufallszahlengenerator aus einer der Klassen DRG.3, DRG.4, PTG.3 oder NTG.1 gemäß [AIS20/31], vgl. auch Kapitel 9 in Teil 1 dieser Technischen Richtlinie [TR-02102-1].

Literaturverzeichnis

ID	Referenz
AIS20/31	BSI: AIS 20/31 – A proposal for: Functionality classes for random number generators, September 2011
BDF14	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
CRIME	J. Rizzo, Th. Duong: The CRIME attack, https://www.ekoparty.org/archive/2012/CRIME_ekoparty2012.pdf, September 2012
IANA	IANA: http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml
LP19	G. Leurent, T. Peyrin: From Collisions to Chosen-Prefix Collisions – Application to Full SHA-1, EUROCRYPT 2019, Lecture Notes in Computer Science, vol. 11478, 2019
RFC2246	T. Dierks, C. Allen: RFC 2246, The TLS Protocol Version 1.0, Januar 1999
RFC4346	T. Dierks, E. Rescorla: RFC 4346, The Transport Layer Security (TLS) Protocol Version 1.1, April 2006
RFC5246	T. Dierks, E. Rescorla: RFC 5246, The Transport Layer Security (TLS) Protocol Version 1.2, August 2008
RFC5289	E. Rescorla: RFC 5289, TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM), August 2008
RFC5487	M. Badra: RFC 5487, Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode, März 2009
RFC5489	M. Badra, I. Hajjeh: RFC 5289, ECDHE_PSK Cipher Suites for Transport Layer Security (TLS), März 2009
RFC5639	M. Lochter, J. Merkle: RFC 5639, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, März 2010
RFC5746	E. Rescorla, M. Ray, S. Dispensa, N. Oskov: RFC 5746, Transport Layer Security (TLS) Renegotiation Indication Extension, Februar 2010
RFC6066	D. Eastlake 3rd: RFC 6066, Transport Layer Security (TLS) Extensions: Extension Definitions, Januar 2011
RFC6176	S. Turner, T. Polk: RFC 6176, Prohibiting Secure Sockets Layer (SSL) Version 2.0, März 2011
RFC6655	D. McGrew, D. Bailey: RFC 6655, AES-CCM Cipher Suites for Transport Layer Security (TLS), Juli 2012
RFC7027	M. Lochter, J. Merkle: RFC 7027, Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS), Oktober 2013
RFC7251	D. McGrew, D. Bailey, M. Campagna, R. Dugal: RFC 7251, AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS, Juni 2014
RFC7465	A. Popov: RFC 7465, Prohibiting RC4 Cipher Suites, Februar 2015
RFC7568	R. Barnes, M. Thomson, A. Pironti, A. Langley: RFC 7568, Deprecating Secure Sockets Layer Version 3.0, Juni 2015
RFC7627	K. Bhargavan, A. Delignat-Lavaud, A. Pironti, A. Langley, M. Ray: RFC 7627, Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension, September 2015
RFC7919	D. Gillmor: RFC 7919, Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS), August 2016
RFC8422	Y. Nir, J. Josefsson, M. Pegourie-Gonnard: RFC 8422, Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier, August 2018
RFC8442	J. Mattsson, D. Migault: RFC 8442, ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2, September 2018
RFC8446	E. Rescorla: RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3, August 2018
RFC8734	L. Bruckert, J. Merkle, M. Lochter: RFC 8734, Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS) Version 1.3, February 2020
SBK17	M. Stevens, E. Bursztein, P. Karpman, A. Albertini, Y. Markov: The first collision for full SHA-1. CRYPTO 2017, Lecture Notes in Computer Science, vol. 10401, 2017

SSLv2	Netscape: Hickman, Kipp: "The SSL Protocol", April 1995
SSLv3	Netscape: A. Frier, P. Karlton, P. Kocher: "The SSL 3.0 Protocol", 1996
TR-02102-1	BSI: Technische Richtlinie TR-02102-1, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, 2020
TR-03116-4	BSI: TR-03116-4, Kryptographische Vorgaben für Projekte der Bundesregierung, Teil 4: Kommunikationsverfahren in Anwendungen, 2020