Federal Office
for Information Security

Technical Guideline TR-02102-2 Cryptographic Mechanisms: Recommendations and Key Lengths

Part 2 – Use of Transport Layer Security (TLS)

Version 2020-01

Document history

Version	Date	Description
2019-01	2019-02-22	Adjustment of the periods of use, recommendation of TLS 1.3, recommendation of PSK cipher suites of RFC 8442, recommendation of CCM mode
2020-01	2020-02-28	Adjustment of the periods of use, discontinuation of HMAC-SHA-1

Federal Office for Information Security
Post Box 20 03 63
D-53133 Bonn
E-Mail: TR02102@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

Tables

1 Introduction

This Technical Guideline contains recommendations for the use of the cryptographic protocol *Transport Layer Security (TLS)*. It is used for the secure transmission of information in data networks, where in particular the confidentiality, integrity and authenticity of the transmitted information are important.

The Technical Guideline at hand contains recommendations for the protocol version to be used as well as the cryptographic algorithms and key lengths as a concretisation of the general recommendations in Part 1 of this Technical Guideline [TR-02102-1]. As mentioned in Part 1 of [TR-02102-1], mechanisms which are not listed are not necessarily considered by the BSI to be insecure.

This Technical Guideline does not contain recommendations for concrete applications, risk assessments or attack vectors that result from errors in the implementation of the protocol.

Note: Even if all recommendations for the use of TLS are being considered, data can leak from a cryptographic system to a considerable amount, e.g. by exploiting side channels (measurement of timing behaviour, power consumption, data rates etc.). Therefore, the developer of a cryptographic system should identify possible side channels by involving experts in this field and implement appropriate countermeasures. Depending on the application, this also applies to fault attacks.

Note: For the definitions of the cryptographic terms used in this document, please see the glossary in [TR-02102-1].

2 Basic information

Transport Layer Security (TLS), formerly known as Secure Socket Layer (SSL), allows the secure transmission of information from the application layer (e.g. HTTPS, FTPS or IMAPS) via TCP/IP-based connections (the Internet in particular).

Before data can be transmitted, a secure connection between the two connection partners (client and server) must be established. This process is called *handshake* and is an important part of the TLS protocol. Client and the server agree upon:

1. Cryptographic algorithms for *data encryption*, *protection of the integrity*, *key agreement* and, if necessary, (one-sided or two-sided) *authentication*. These algorithms are defined by the cipher suite and further cryptographic parameters (see Sections 3.3 and 3.4).
2. A shared secret, the *master secret*, from which the session keys for the protection of the integrity and for encryption will be derived.

Note: The TLS protocol also allows for connections that are not authenticated or authenticated only on one side (example: HTTPS connections are usually authenticated only on the server side). For this reason, system developers of cryptographic systems should take into consideration whether further authentication in the application layer is required (example: authentication of a home banking user by requiring a password). For particularly critical operations, authentication by means of knowledge and ownership (two-factor authentication) should be carried out in general. Such authentication should also cover the transmitted data by using cryptographic mechanisms.

3 Recommendations

3.1 General remarks

3.1.1 Periods of use

The recommendations in this Technical Guideline have a maximum period of use. The indication of the year means that the respective mechanism can be used until the end of the year stated. If the year is marked with a "+" sign, it is possible to extend the period of use.

3.1.2 Security level

The security level for all cryptographic mechanisms in this Technical Guideline depends on the security level stated in Section 1.1 in [TR-02102-1]. The current security level is 100 bits.

Note: It is planned to raise the security level to 120 bits from 2023. As an interim arrangement, the usage of RSA-based signature and encryption algorithms with a key size of at least 2000 bits will however remain compliant with this Technical Guideline through 2023. See also Section 1.1 in [TR-02102-1].

3.1.3 Key lengths for EC algorithms

Until the end of 2022 the security level for algorithms which are based on elliptic curves (EC) has been chosen slightly larger (compared to RSA) in this Technical Guideline in order to reach a security margin for the EC algorithms (see Section 3.6). For reasons and further explanations, see Remark 4 in Chapter 3 in [TR-02102-1].

3.2 SSL/TLS versions

The SSL protocol is available in the versions 1.0, 2.0 and 3.0, whereby version 1.0 was not published. TLS 1.0 is a direct further development of SSL 3.0 and is specified in [RFC2246]. Furthermore, the TLS protocol is available in the versions 1.1, 1.2, and 1.3 which are specified in [RFC4346], [RFC5246], and [RFC8446].

Recommendations for the choice of the TLS version:

- In general, TLS 1.2 or TLS 1.3 should be used.
- TLS 1.0 and TLS 1.1 are **not recommended** (see also Section 3.3.1.3).
- SSL v2 ([SSLv2]) and SSL v3 ([SSLv3]) are **not recommended** (see also [RFC6176] and [RFC7568]).

3.3 Recommendations for TLS 1.2

In TLS 1.2, cryptographic mechanisms of a connection are defined by a cipher suite. A cipher suite specifies a key agreement mechanism (with authentication) for the handshake protocol, an authenticated encryption algorithm for the record protocol, and a hash function for key derivation. Depending on the cipher suite, a Diffie-Hellman group (in a finite field or over an elliptic curve) and a signature algorithm for key agreement must be specified as well.

A complete list of all defined cipher suites with references to the respective specifications is available at [IANA].

3.3.1 Cipher suites

For cipher suites in TLS 1.2, the naming convention `TLS_`*`AKE`*`_WITH_`*`Enc`*`_`*`Hash`* is usually used, where *`AKE`* denotes a key agreement mechanism (with authentication), *`Enc`* denotes an encryption algorithm with mode of operation, and *`Hash`* denotes a hash function. The function *`Hash`* is used for an HMAC (*Keyed-Hash Message Authentication Code*) which is employed by the PRF (*Pseudo-Random Function*) used for key derivation.[1] If *`Enc`* is not an AEAD algorithm (*Authenticated Encryption with Associated Data*), then HMAC is also utilized for integrity protection in the record protocol.

In general, it is recommended to only use cipher suites which meet the requirements for algorithms and key lengths as given in [TR-02102-1].

3.3.1.1 (EC)DHE cipher suites

The use of the following cipher suites with Perfect Forward Secrecy[2] is recommended:

1 For cipher suites using the CCM mode of operation, no hash function is indicated. These cipher suites use SHA-256 for the PRF.

2 Perfect Forward Secrecy (abbreviated PFS, also Forward Secrecy) means that a connection cannot be decrypted subsequently even if the long-term keys of the communication partners are known. When using TLS in order to protect personal or other sensitive data, Perfect Forward Secrecy is generally recommended.

Cipher suite	IANA no.	Specified in	Use up to
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+
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+

Table 1: Recommended cipher suites for TLS 1.2 with Perfect Forward Secrecy

3.3.1.2 (EC)DH cipher suites

If the use of the cipher suites with Perfect Forward Secrecy recommended in Section 3.3.1.1 is not possible, the following cipher suites can also be used:

Cipher suite	IANA no.	Specified in	Use up to
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+

Cipher suite	IANA no.	Specified in	Use up to
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+

Table 2: Recommended cipher suites for TLS 1.2 without Perfect Forward Secrecy

3.3.1.3 Key agreement with pre-shared data

If additional data that have been exchanged in advance are to be incorporated into the key agreement, cipher suites with a pre-shared key (abbreviated PSK) can be used. Generally, it is recommended to use cipher suites for which further ephemeral keys or previously exchanged random numbers are incorporated into the key agreement in addition to the pre-shared key.

Using cipher suites of type `TLS_PSK_*`, i.e. without additional ephemeral keys or random numbers, is **not recommended**, because the security of the connection is based solely on the entropy and confidentiality of the pre-shared keys for these cipher suites.

The use of the following cipher suites with PSK is recommended:

Cipher suite	IANA no.	Specified in	Use up to
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+

Table 3: Recommended cipher suites for TLS 1.2 with pre-shared key

Note: The cipher suites `TLS_RSA_PSK_*` in Table 3 do *not* provide Perfect Forward Secrecy, whereas all other cipher suites from Table 3 do provide Perfect Forward Secrecy.

3.3.1.4 Interim arrangements

SHA-1 is not a collision resistant hash function; while the generation of collisions for SHA-1 takes considerable effort, it is nonetheless doable in practice [SBK17, LP19]. From a security-technical perspective, however, there is, according to present knowledge, no reason speaking against using it in constructions which do not require collision resistance (for example, as a basis for an HMAC or as a component of a pseudo-random number generator). It is recommended to use a hash function of the SHA-2 family or of the SHA-3 family also in these applications as a general security safeguard. In principle, using SHA-1 in the HMAC construction or in other cryptographic mechanisms with comparable cryptographic requirements for the hash function used (for example, within the framework of a pseudo-random number generator or as a part of the mask generation function in RSA-OAEP) is in conformity with this Technical Guideline until 2019.

Deviating from the recommendations in Part 1 of this Technical Guideline, SHA-1 (i.e. cipher suites of the form `*_SHA`) can still be used on an interim basis in *existing* applications as a hash function for the protection of the integrity as part of HMAC. Irrespective of the *maximum* period of use stated in Table 4, migrating to SHA-256 or SHA-384 and TLS 1.2 as quickly as possible is recommended.

Note: Since TLS 1.1 uses the hash function SHA-1 as a component for the creation of signatures (and does not support the SHA-2 family), this protocol version is no longer recommended after the discontinuation of SHA-1.

The encryption algorithm RC4 in TLS has significant security weaknesses. Using it is therefore not recommended (see also [RFC7465]).

Deviation	**Use maximally up to**	**Recommendation**
SHA-1 for HMAC calculation and as a component of the PRF in TLS	2019	Migration to SHA-256/-384
SHA-1 as a component for the generation of signatures in TLS	2015	Migration to SHA-256/-384/-512

Table 4: Interim arrangements

Note: In this Technical Guideline, SHA-1 as a component for the creation of signatures was only recommended in the context of TLS (see Table 4) in order to allow the use of TLS 1.0 on an interim basis until the end of 2015. Since SHA-1 is required for the handshake in TLS 1.0 and since a SHA-1 collision can presumably not be computed in real-time (during the handshake), the use of SHA-1 was allowed somewhat longer in this special case.

Generally, the use of SHA-1 (for instance for the creation of signatures) is not recommended since 2013 in TR-02102-1.

3.3.2 Diffie-Hellman groups

For cipher suites of type `TLS_DHE_*` or `TLS_ECDHE_*`, the client can use the extension “supported_groups” (formerly also called “elliptic_curves”) to inform the server about the Diffie-Hellman groups he wants to use (see [RFC7919] for DHE and [RFC8422] for ECDHE).

The use of the extension “supported_groups” for `TLS_ECDHE_*` cipher suites is recommended.

The use of the extension “supported_groups” for `TLS_DHE_*` cipher suites is recommended as soon as suitable implementations are available.

The use of the following Diffie Hellman groups is recommended:

Diffie-Hellman group	IANA no.	Specified in	Use up to
`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+

Table 5: Recommended Diffie-Hellman groups for TLS 1.2

In general, Section 3.6 has to be taken into consideration for the choice of domain parameters and key lengths.

3.3.3 Signature algorithms

In TLS 1.2, the client can use the extension “signature_algorithms” (see [RFC5246]) to inform the server about the signature algorithms he wants to use for key agreement and certificates. The algorithm has to be specified as combination of signature algorithm and hash function.

The use of the extension “signature_algorithms” is recommended.

The use of the following signature algorithms is recommended:

Signature algorithm	IANA no.	Specified in	Use up to
`rsa`	1	[RFC5246]	2025
`dsa`	2	[RFC5246]	2026+
`ecdsa`	3	[RFC5246]	2026+

Table 6: Recommended signature algorithms for TLS 1.2

For domain parameters and key lengths Section 3.6 has to be taken into consideration.

Note: The use of the signature algorithm `rsa` (IANA no. 1) is recommended only up to 2025, because it uses the PKCS #1 v1.5 padding scheme (see also Section 1.4 in [TR-02102-1]).

The use of the following hash functions (combined with a signature algorithm in Table 6) is recommended:

Hash function	IANA no.	Specified in	Use up to
`sha256`	4	[RFC5246]	2026+
`sha384`	5	[RFC5246]	2026+
`sha512`	6	[RFC5246]	2026+

Table 7: Recommended hash functions for signature algorithms in TLS 1.2

3.3.4 Further recommendations

3.3.4.1 Session renegotiation

It is recommended to use *session renegotiation* only on the basis of [RFC5746]. Renegotiation initiated by the client should by rejected by the server.

3.3.4.2 Truncated HMAC output

The extension "truncated_hmac" defined in [RFC6066] to truncate the HMAC output to 80 bits should *not* be used.

3.3.4.3 TLS compression and the CRIME attack

TLS offers the option of compressing the transmitted data prior to encryption. This results in a possible side-channel attack on the encryption by exploiting the length of the encrypted data (see [CRIME]).

In order to prevent this, it must be ensured that all data of a data packet come from correct and legitimate connection partners and that the attacker cannot perform a plaintext injection. If this cannot be ensured, it is recommended not to use TLS data compression.

3.3.4.4 The Lucky 13 attack

Lucky 13 is a side-channel attack (timing) against CBC-mode cipher suites, in which the attacker exploits very small time differences when processing the padding on the server. For this attack, it is necessary that the attacker can measure the time in the network very accurately. The attacker sends manipulated cipher texts to the server and measures the time which the server takes to check the padding or to report an error. The network jitter, however, can very easily lead to measurement errors when measuring the time so that an attack generally appears to be difficult, since the attacker in the network has to be “very close” to the server in order to be able to measure sufficiently accurately.

The attack can also be fended off if

- authenticated encryption, such as AES-GCM or AES-CCM, or
- encrypt-then-MAC (see also following Section)

is used.

3.3.4.5 The Encrypt-then-MAC extension

According to the TLS specification (see [RFC5246]), the transmitted data are protected with a Message Authentication Code (MAC) first and then provided with a padding; afterwards, the data and the padding are encrypted. In the past, this order (“MAC-then-encrypt”) has been a common reason for attacks on the encryption, since the padding is not protected by the MAC.

In the case of so-called padding oracle attacks, the encrypted TLS packets are manipulated by a man-in-the-middle attacker in order to exploit the verification of the padding as a side channel. For example, this may lead to an attacker being able to decrypt an HTTPS session cookie and thus take over the session of the victim.

[RFC7366] specifies the TLS extension “encrypt-then-MAC”. Here, the data to be transmitted are provided with a padding first, then encrypted and protected with an MAC afterwards. Thus, manipulations of the padding are impossible, since it is also protected by the MAC.

The use of the TLS extension "encrypt-then-MAC" according to [RFC7366] will be recommended as soon as suitable implementations are available.

3.3.4.6 The Heartbeat extension

The Heartbeat extension is specified in [RFC6520]. It allows to maintain a TLS connection over a longer period of time without having to perform a renegotiation of the connection. Due to the so-called Heartbleed bug, the attacker is able to access certain memory areas of the server which might contain secret key material. This may result in a complete compromise of the server if the private key of the server becomes known.

Recommendation: It is urgently recommended not to use the Heartbeat extension. If it is still necessary, it should be ensured that the TLS implementation is not susceptible to the Heartbleed bug.

3.3.4.7 The Extended Master Secret extension

In order to fend off attacks such as the triple handshake attack (see [BDF14]), it is very useful to incorporate further connection parameters into the TLS handshake to ensure that different TLS connections also use different master secrets (from which the symmetric keys are derived).

[RFC7627] specifies the TLS extension *Extended Master Secret* which incorporates a hash value over all messages of the TLS handshake when calculating the "extended" master secret.

Using the TLS extension *Extended Master Secret* according to [RFC7627] is recommended as soon as suitable implementations are available.

3.4 Recommendations for TLS 1.3

In TLS 1.3, the cryptographic mechanisms of a connection are defined by a handshake mode, a Diffie-Hellman group (if (EC)DHE is used), a signature algorithm (if certificate-based authentication is used), and a cipher suite. In contrast to earlier versions of TLS, a cipher suite specifies only an authenticated encryption algorithm for the record protocol and a hash function for key derivation.

3.4.1 Handshake modes

Besides Diffie-Hellman key agreement over finite fields (DHE) or elliptic curves (ECDHE), TLS 1.3 offers additional handshake modes using pre-shared keys (PSK). In this context, pre-shared keys either refer to keys which are provisioned out-of-band or to key material that has been established in a previous session via the session ticket mechanism.

The use of the following PSK modes is recommended:

PSK mode	IANA no.	Specified in	Use up to
`psk_ke`	0	[RFC8446]	2026+
`psk_dhe_ke`	1	[RFC8446]	2026+

Table 8: Recommended pre-shared key modes for TLS 1.3

Note: The PSK mode `psk_ke` does not offer Perfect Forward Secrecy. This mode should only be used in special applications after consultation of an expert.

Note: TLS 1.3 offers an option to include application data already in the first message of a PSK handshake (*zero round-trip time* data, abbreviated 0-RTT data). This data is not protected against replay attacks. Therefore, sending or accepting 0-RTT data is **not recommended**.

3.4.2 Diffie-Hellman groups

In TLS 1.3, client and server can use the extension "supported_groups" to inform each other about the Diffie-Hellman groups they want to use for (EC)DHE.

The use of the following Diffie-Hellman groups is recommended:

Diffie-Hellman group	IANA no.	Specified in	Use up to
`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+

Table 9: Recommended Diffie-Hellman groups for TLS 1.3

Note: In general, the Brainpool curves are recommended.

Note: In [RFC8446], the IANA numbers of some EC groups, that are either obsolete or have had little usage according to [RFC8446], have been marked as "obsolete_RESERVED". Among those are the IANA numbers 26, 27, 28, which are allocated for the Brainpool curves for usage in TLS 1.2 and earlier TLS versions. For using the Brainpool curves in TLS 1.3, the IANA numbers 31, 32, 33 have been allocated (see [RFC8734]).

3.4.3 Signature algorithms

In TLS 1.3, client and server can use the extensions "signature_algorithms" and "signature_algorithms_cert" to inform each other about the signature algorithms they want to use for certificate-based authentication. The extension "signature_algorithms" refers to signatures which are generated by client or server for their CertificateVerify message and the extension "signature_algorithms_cert" refers to signatures in certificates.

The use of the following signature algorithms for the extension "signature_algorithms" is recommended:

Signature algorithm	IANA no.	Specified in	Use up to
`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+

Table 10: Recommended signature algorithms for TLS 1.3 (client/server signatures)

The use of the following signature algorithms for the extension “signature_algorithms_cert” is recommended:

Signature algorithm	IANA no.	Specified in	Use up to
`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+

Table 11: Recommended signature algorithms for TLS 1.3 (signatures in certificates)

For key lengths of RSA-signatures, Section 3.6 has to be taken into consideration.

Note: The use of the signature algorithms `rsa_pkcs1_*` (IANA no. 0x0401, 0x0501, and 0x0601) is recommended only up to 2025, because they use the PKCS #1 v1.5 padding scheme (see also Section 1.4 in [TR-02102-1]).

3.4.4 Cipher suites

For cipher suites in TLS 1.3, the naming convention `TLS_`*`AEAD`*`_`*`Hash`* is used, where *`AEAD`* denotes an authenticated encryption algorithm (*authenticated encryption with associated data*, abbreviated AEAD) for the record protocol and *Hash* denotes a hash function for usage with HMAC (*Keyed-Hash Message Authentication Code*) and HKDF (*HMAC-based Extract-and-Expand Key Derivation Function*) in the handshake protocol.

The use of the following cipher suites is recommended:

Cipher suite	IANA no.	Specified in	Use up to
`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+

Table 12: Recommended cipher suites for TLS 1.3

3.5 Authentication of the communication partners

The TLS protocol offers the following three ways of authenticating the communication partners:

- Authentication of both communication partners
- Authentication on the server side only
- No authentication

The necessity of authentication depends on the application. When using TLS on the web, at least an authentication of the server is generally necessary. When using TLS in closed systems (VPN or the like), authentication on both sides is usually required.

For authentication within Federal Government projects, the requirements of Technical Guideline [TR-03116-4] in its currently valid version must be taken into account.

3.6 Domain parameters and key lengths

The domain parameters and key lengths for

- static key pairs of the communication partners,
- ephemeral key pairs when using cipher suites with Forward Secrecy, and
- key pairs for the signature of certificates

must comply with the recommendations in Part 1 of this Technical Guideline [TR-02102-1].

3.6.1 Key lengths

It is recommended to use at least the following key lengths:

Algorithm	Minimum key length	Use from (at the latest)	Use up to
Signature keys for certificates and key agreement			
ECDSA	224 bits		2015
	250 bits		2026+
DSS	2000 bits		2022
	3000 bits	2023	2026+
RSA	2000 bits		2023
	3000 bits	2024	2026+
Static und ephemeral Diffie-Hellman keys			
ECDH	224 bits		2015
	250 bits		2026+
DH	2000 bits		2022
	3000 bits	2023	2026+

Table 13: Recommended minimum key lenghts for the TLS handshake protocol

Note: If a key pair is *static*, it is reused several times for new connections. In contrast to this, *ephemeral* means that a new key pair is created and used for each new connection. Ephemeral keys must be deleted securely after the connection is terminated, see Section 4.2. If the connection shall provide Perfect Forward Secrecy, then only ephemeral keys must be used.

Important Note: It is reasonable to use a key length of 3000 bits for RSA, DH, and DSS, in order to achieve a homogenous security level for all asymmetric mechanisms. From 2023, a key length of at least 3000 bits is required for cryptographic implementations that shall comply with this Technical Guideline.

Systems with a lifetime up to 2022 that use a key length of at least 2000 bits are compliant with this Technical Guideline. As an interim arrangement, the usage of RSA keys of length at least 2000 bits will remain compliant up to 2023. This is the recommended minimum key length for RSA, DH, and DSS. For further information see Remarks 4 and 5 in Chapter 3 in [TR-02102-1].

Note: The recommendations in this Technical Guideline are suitable to reach the security level stated in Section 3.1.2, which is at the moment 100 bits.

The prediction period for the recommendations at hand is 7 years. Appropriate recommendations for larger periods, as they can be found in other publicly available documents, are naturally very hard to make because future cryptographic developments cannot be predicted precisely for larger periods. In such cases, these recommendations contain parameters and key lengths that might exceed those given in this Technical Guideline.

3.6.2 Use of elliptic curves

When using elliptic curves, cryptographically strong curves over finite fields of the form F_p (p prime) are always recommended. In addition, it is recommended to only use *named curves* (see Section "Supported Groups Registry" in [IANA]) in order to avoid attacks via unverified weak domain parameters. The following named curves are recommended:

- brainpoolP256r1, brainpoolP384r1, brainpoolP512r1 (see [RFC5639] and [RFC7027])

If these curves are not available, the following curves can also be used:

- secp256r1, secp384r1

4 Keys and random numbers

4.1 Key storage

Private cryptographic keys, especially static keys and signature keys, must be stored and processed in a secure manner. This includes, among other things, the protection against copying, misuse and manipulation of the keys. Secure storage of the keys can be achieved, for example by using certified hardware (chip card, HSM).

The public keys of trusted bodies (trust anchors) must also be stored in such a manner that they cannot be manipulated.

4.2 Handling of ephemeral keys

If a cipher suite with Perfect Forward Secrecy is used, it should be ensured that all ephemeral keys are deleted irrevocably after they have been used and that no copies of these keys were made. Ephemeral or session keys should only be used for *one* connection and generally not be stored persistently.

4.3 Random numbers

For the generation of random numbers, for example for cryptographic keys or for creating signatures, appropriate random number generators must be used.

A random number generator from one of the classes DRG.3, DRG.4, PTG.3 or NTG.1 according to [AIS20/31] is recommended, see also Chapter 9 in Part 1 of this Technical Guideline [TR-02102-1].

References

ID	Reference
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, January 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, March 2009
RFC5489	M. Badra, I. Hajjeh: RFC 5289, ECDHE_PSK Cipher Suites for Transport Layer Security (TLS), March 2009
RFC5639	M. Lochter, J. Merkle: RFC 5639, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, March 2010
RFC5746	E. Rescorla, M. Ray, S. Dispensa, N. Oskov: RFC 5746, Transport Layer Security (TLS) Renegotiation Indication Extension, February 2010
RFC6066	D. Eastlake 3rd: RFC 6066, Transport Layer Security (TLS) Extensions: Extension Definitions, January 2011
RFC6176	S. Turner, T. Polk: RFC 6176, Prohibiting Secure Sockets Layer (SSL) Version 2.0, March 2011
RFC6655	D. McGrew, D. Bailey: RFC 6655, AES-CCM Cipher Suites for Transport Layer Security (TLS), July 2012
RFC7027	M. Lochter, J. Merkle: RFC 7027, Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS), October 2013
RFC7251	D. McGrew, D. Bailey, M. Campagna, R. Dugal: RFC 7251, AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS, June 2014
RFC7465	A. Popov: RFC 7465, Prohibiting RC4 Cipher Suites, February 2015
RFC7568	R. Barnes, M. Thomson, A. Pironti, A. Langley: RFC 7568, Deprecating Secure Sockets Layer Version 3.0, June 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