Federal Office
for Information Security

Technical Guideline TR-02102-4 Cryptographic Mechanisms: Recommendations and Key Lengths

Part 4 – Use of Secure Shell (SSH)

Document history

Table 1: Document history

Version	***Date***	***Description***
2019-01	2019-02-22	Adjustment of the periods of use, recommendations of Diffie-Hellman groups of RFC 8268
2020-01	2020-01-31	Adjustment of the periods of use, discontinuation of HMAC-SHA-1
2021-01	2021-03-12	Adjustment of the periods of use
2022-01	2022-01-24	Adjustment of the periods of use
2023-01	2023-01-17	Increase of the security level to 120 bits, adjustment of the periods of use
2024-01	2024-02-29	Adjustment of the periods of use, discontinuation of DSA recommendation from 2029

Federal Office for Information Security
P.O. Box 20 03 63
53133 Bonn
Tel.: +49 22899 9582-0
E-Mail: TR02102@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

1 Introduction

This Technical Guideline provides recommendations for the use of the cryptographic protocol *Secure Shell (SSH)*. This protocol can be used to establish a secure channel within an insecure network. The most common applications of the SSH protocol are logging on to a remote system (remote command-line login) and executing commands or running applications on remote systems.

This Technical Guideline contains recommendations for the protocol version to be used and the cryptographic algorithms as a concretisation of the general recommendations in Part 1 of this Technical Guideline (see [TR-02102-1]).

This Technical Guideline does not contain recommendations for specific applications, risk assessments or points of attack, which result from errors in the implementation of the protocol.

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

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

2 Basic information

The SSH protocol consists of the three subprotocols *Transport Layer Protocol*, *User Authentication Protocol* and *Connection Protocol*.

The Transport Layer Protocol (see [RFC 4253]) allows server authentication, encryption, protection of the integrity and, optionally, data compression. It is based logically on the TCP/IP protocol.

The User Authentication Protocol (see [RFC 4252]) is used to authenticate the user to the server. It is based on the Transport Layer Protocol.

The Connection Protocol (see [RFC 4254]) is responsible for creating and managing logical channels within the encrypted tunnel. It is based on the User Authentication Protocol.

For more detailed information about the protocol architecture of SSH, see [RFC 4251].

The comprehensive specification of the SSH-2 (see Section 3.2) protocol can be found in the following RFCs:

- RFC 4250: The Secure Shell (SSH) Protocol Assigned Numbers (January 2006)
- RFC 4251: The Secure Shell (SSH) Protocol Architecture (January 2006)
- RFC 4252: The Secure Shell (SSH) Authentication Protocol (January 2006)
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol (January 2006)
- RFC 4254: The Secure Shell (SSH) Connection Protocol (January 2006)
- RFC 4256: Generic Message Exchange Authentication for the Secure Shell Protocol (SSH) (January 2006)
- RFC 4335: The Secure Shell (SSH) Session Channel Break Extension (January 2006)
- RFC 4344: The Secure Shell (SSH) Transport Layer Encryption Modes (January 2006)

The following RFCs contain extensions and additions to the SSH protocol:

- RFC 4255: Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints (January 2006)
- RFC 4419: Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4432: RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4462: Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol (May 2006)
- RFC 4716: The Secure Shell (SSH) Public Key File Format (November 2006)
- RFC 4819: Secure Shell Public Key Subsystem (March 2007)
- RFC 5647: AES Galois Counter Mode for the Secure Shell Transport Layer Protocol (August 2009)
- RFC 5656: Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer (December 2009)
- RFC 6187: X.509v3 Certificates for Secure Shell Authentication (March 2011)
- RFC 6239: Suite B Cryptographic Suites for Secure Shell (SSH) (May 2011)
- RFC 6594: Use of the SHA-256 Algorithm with RSA: Digital Signature Algorithm (DSA): and Elliptic Curve DSA (ECDSA) in SSHFP Resource Records (April 2012)
- RFC 6668: SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol (July 2012)
- RFC 8268: More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH) (December 2017)

- RFC 8270: Increase the Secure Shell Minimum Recommended Diffie-Hellman Modulus Size to 2048 Bits (December 2017)
- RFC 9142: Key Exchange (KEX) Method Update and Recommendations for Secure Shell (SSH) (January 2022)

3 Recommendations

This chapter provides recommendations for the use of the SSH protocol. They refer to the cryptographic mechanisms to be used and the SSH versions. This Technical Guideline does not contain configuration instructions for specific implementations of the SSH protocol, but general cryptographic recommendations that can be used for all SSH implementations.

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] and is 120 bits.

3.2 SSH versions

In 1995, the first version of the SSH protocol was developed by Tatu Ylönen, a researcher at Helsinki University of Technology. Today, this version is referred to as SSH-1. In 2006, a revised version of the protocol was adopted by the IETF as an internet standard (RFC). This version is referred to as SSH-2.

The following recommendations apply to the choice of the SSH protocol version:

- The use of SSH-2 is recommended.
- Using SSH-1 is **not recommended**, since this protocol version contains cryptographic vulnerabilities.

3.2.1 Conformity to the SSH specification

The specification of the SSH protocol contains cryptographic algorithms which must be supported by applications conforming to the standard. In this Technical Guideline, not all of them may be recommended.

If an application has to fully comply with the SSH specification, the following procedure should be used: The application should support the algorithms recommended in this document and the algorithms given in the specification, but be configured in such a way that the recommended (cryptographically strong) algorithms are used with high priority and the (possibly cryptographically weak or obsolete) algorithms from the SSH specification are used with low priority or, if possible, are not used.

3.3 Key agreement

When establishing the SSH connection, keys are exchanged in order to create and exchange shared session keys for authentication and encryption.

The following key exchange methods are recommended:

Table 2: Recommended key exchange methods

No.	*Key exchange method*	*Specification*	*Use up to*
1	`diffie-hellman-group-exchange-sha256`	[RFC 4419], Section 4.2 [RFC 8270]	2030+
2	`diffie-hellman-group15-sha512`	[RFC 8268]	2030+

No.	*Key exchange method*	*Specification*	*Use up to*
3	`diffie-hellman-group16-sha512`	[RFC 8268]	2030+
4	`ecdh-sha2-nistp256`	[RFC 5656]	2030+
5	`ecdh-sha2-nistp384`	[RFC 5656]	2030+
6	`ecdh-sha2-nistp521`	[RFC 5656]	2030+

Remark on key exchange method no. 1: Using the notation from Chapter 3 in [RFC4419]:

1. The length of the prime number p should be at least 3000 bits (see also Section 2.3.5 in [TR-02102-1]).
2. The order of the generator g should at least have the size 2^{250} (see also Section 2.3.5 in [TR-02102-1]).
3. According to Chapter 3 in [RFC 4419], p should be a *safe prime*, which means that with $p = 2q+1$, both p and q are prime.

Since p is a *safe prime*, we have $p-1 = 2q$, which means that the order of the generator g can only be 2 or q. If the recommendation in Item 2 is taken into account, only q remains as the possible order. Due to the additional requirement in Item 3 (safe prime), the bit length of q is much greater than the minimum recommendation in Item 2. This circumstance must be accepted if the implementation is to comply with [RFC 4419], [TR-02102-1] and this Technical Guideline.

Note: For this key exchange method, SHA-256 must also be used for the key derivation pseudo-random function (PRF).

Remark on key exchange methods no. 4-6: The corresponding hash function from the SHA-2 family must be chosen depending on the bit length of the curve according to Section 6.2.1 in [RFC5656].

3.3.1 Key re-exchange

It makes sense to renew the key material of a connection after a certain period of time or a certain amount of data transmitted in order to make it more difficult to attack the session keys. With SSH, it is possible to renew the session keys by sending the message `SSH_MSG_KEXINIT`. Both the client and the server can initiate this process.

It is recommended to re-exchange the keys according to Chapter 9 in [RFC 4253], i.e. the session keys are renewed after one hour or after one gigabyte has been transmitted (whichever happens first).

3.4 Encryption algorithms

During the key exchange, the client and the server agree on an encryption algorithm as well as a shared encryption key. The following encryption algorithms are recommended in this respect:

Table 3: Recommended encryption algorithms

No.	*Encryption algorithm*	*Specification*	*Use up to*
1	`AEAD_AES_128_GCM`	[RFC 5647], Section 6.1	2030+
2	`AEAD_AES_256_GCM`	[RFC 5647], Section 6.2	2030+
3	`aes128-ctr`	[RFC 4344]	2030+
4	`aes192-ctr`	[RFC 4344]	2030+
5	`aes256-ctr`	[RFC 4344]	2030+

Note: Algorithm no. 1 and no. 2 already include MAC protection by the GCM mode.

3.5 MAC protection

For the MAC protection, the following algorithms are recommended:

Table 4: Recommended algorithms for MAC protection

No.	MAC algorithm	Specification	Use up to
1	`hmac-sha2-256`	[RFC 6668], Chapter 2	2030+
2	`hmac-sha2-512`	[RFC 6668], Chapter 2	2030+

3.6 Server authentication

The server authenticates to the client. This process is carried out as part of the Transport Layer Protocol. Chapter 7 in [RFC 4253] describes the *explicit server authentication*. The key exchange messages contain a digital signature of the server (or other proof) in order to prove its authenticity. The client can check the signature with the public key of the server and thus determine the authenticity of the server.

The algorithms for digital signatures are called "Public Key Algorithms" in [RFC 4253] (see Section 6.6 in [RFC 4253]).

The following signature algorithms are recommended for server authentication:

Table 5: Recommended signature algorithms for sever authentication

No.	Signature algorithm	Specification	Use up to
1	`pgp-sign-dss`	[RFC 4253], Section 6.6	2029
2	`ecdsa-sha2-*`	[RFC 5656]	2030+
3	`x509v3-ecdsa-sha2-*`	[RFC 6187]	2030+

Remark on signature algorithm no. 1: Using the notation from Section 13.6 in [RFC 4880], the minimum recommended length of the prime p is 3000 bits and the minimum recommended size of the order q is 250 bits (see also Section 2.3.5 in [TR-02102-1]). As corresponding hash function, SHA-256, SHA-384, or SHA-512 is recommended.

The use of the signature algorithm `pgp-sign-dss` is recommended only up to 2029, because it is not widely employed and no longer approved in [FIPS 186-5] (see also Remark 5.7 in [TR-02102-1]).

Remark on signature algorithms no. 2-3: The "`*`" symbol is replaced by the identifier of an elliptic curve from Section 10.1 in [RFC 5656]. The following elliptic curves are currently recommended:

- `nistp256`, `nistp384`, `nistp521`

The corresponding hash function from the SHA-2 family must be chosen depending on the bit length of the curve according to Section 6.2.1 in [RFC5656].

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

3.7 Client authentication

Client authentication (unlike server authentication) does not take place in the Transport Layer Protocol, but in the User Authentication Protocol. This protocol is logically based on the Transport Layer Protocol.

The most important methods for client authentication include:

- Public key authentication
- Password authentication
- Host-based authentication

Recommendation: For client authentication, "public key authentication" together with one of the methods from Table 5, Section 3.6 is recommended.

Remark: According to [RFC 4252], public key authentication must be supported by all SSH implementations. The corresponding Authentication Method Name according to [RFC 4250], Section 4.8 is referred to as "publickey". The authentication method is described in Chapter 7 of [RFC 4242].

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

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 an SSH connection is secured by means of an encryption algorithm, it must 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 may 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 [AIS 20/31] is recommended, see also Chapter 8 in Part 1 of this Technical Guideline [TR-02102-1].

Bibliography

[**AIS 20/31**] BSI: AIS 20/31 – A proposal for: Functionality classes for random number generators, 2011

[**FIPS 186-5**] National Institute of Standards and Technology: Federal Information Processing Standards FIPS PUB 186-5, Digital Signature Standard (DSS), 2023

[**RFC 4251**] T. Ylonen, C. Lonvick: RFC 4251, The Secure Shell (SSH) Protocol Architecture, 2006

[**RFC 4252**] T. Ylonen, C. Lonvick: RFC 4252, The Secure Shell (SSH) Authentication Protocol, 2006

[**RFC 4253**] T. Ylonen, C. Lonvick: RFC 4253, The Secure Shell (SSH) Transport Layer Protocol, 2006

[**RFC 4254**] T. Ylonen, C. Lonvick: RFC 4254, The Secure Shell (SSH) Connection Protocol, 2006

[**RFC 4344**] M. Bellare, T. Kohno, C. Namprempre: RFC 4344, The Secure Shell (SSH) Transport Layer Encryption Modes, 2006

[**RFC 4419**] M. Friedl, N. Provos, W. Simpson: RFC 4419, Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol, 2006

[**RFC 4432**] B. Harris: RFC 4432, RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol, 2006

[**RFC 4880**] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer: RFC 4880, OpenPGP Message Format, 2007

[**RFC 5647**] K. Igoe, J. Solinas: RFC 5647, AES Galois Counter Mode for the Secure Shell Transport Layer Protocol, 2009

[**RFC 5656**] D. Stebila, J. Green: RFC 5656, Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer, 2009

[**RFC 6668**] D. Bider, M. Baushke: RFC 6668, SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol, 2012

[**RFC 8268**] M. Baushke: RFC 8268, More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH), 2017

[**RFC 8270**] L. Velvindron, M. Baushke: RFC 8270, Increase the Secure Shell Minimum Recommended Diffie-Hellman Modulus Size to 2048 Bits, 2017

[**RFC 9142**] M. Baushke: RFC 9142, Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH), 2022

[**TR-02102-1**] BSI: Technische Richtlinie TR-02102-1, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, 2024

[**TR-03116-4**] BSI: Technische Richtlinie TR-03116-4, Kryptographische Vorgaben für Projekte der Bundesregierung, Teil 4: Kommunikationsverfahren in Anwendungen, 2024