Workgroup: Internet Engineering Task Force
Internet-Draft:
draft-ietf-curdle-ssh-kex-sha2-13
Updates: 4250 4253 4432 4462 (if approved)
Published: 14 January 2021
Intended Status: Standards Track
Expires: 18 July 2021
Authors: M. D. Baushke
Juniper Networks, Inc.

Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH)

Abstract

This document is intended to update the recommended set of key exchange methods for use in the Secure Shell (SSH) protocol to meet evolving needs for stronger security. This document updates RFC 4250, RFC 4253, RFC 4432, and RFC 4462.

Status of This Memo

Copyright Notice

Table of Contents

1. Overview and Rationale

Secure Shell (SSH) is a common protocol for secure communication on the Internet. In [RFC4253], SSH originally defined two Key Exchange (KEX) Method Names that MUST be implemented. Over time what was once considered secure is no longer considered secure. The purpose of this RFC is to recommend that some published key exchanges be deprecated as well as recommending some that SHOULD and one that MUST be adopted. This document updates [RFC4250] [RFC4253] [RFC4432] [RFC4462] by changing the requirement level ("MUST" moving to "SHOULD" or "MAY" or "SHOULD NOT", and "MAY" moving to "MUST" or "SHOULD" or "SHOULD NOT" or "MUST NOT") of various key-exchange mechanisms.

A key exchange has two components, a hashing algorithm and a public key algorithm. The following subsections describe how to select each component.

1.1. Selecting an appropriate hashing algorithm

The SHA-1 hash is in the process of being deprecated for many reasons. There have been attacks against SHA-1 that have shown there are weaknesses in the algorithm. Therefore, it is desirable to move away from using it before attacks become more serious.

At present, the attacks against SHA-1 are collision attacks that usually rely on human help rather than a pre-image attack. SHA-1 resistance against 2nd pre-image is still at 160 bits, but SSH does not depend on that, but rather on chosen prefix resistance.

Transcript Collision attacks are documented in [TRANS-COLL]. This paper shows that the man in the middle does not tamper with the Diffie-Hellman values and does not know the connection keys. However, it manages to tamper with both Ic and Is, and can therefore downgrade the negotiated ciphersuite to a weak cryptographic algorithm that the attacker knows how to break.

These attacks are still computationally very difficult to perform, but is is desirable that any Key Exchanging using SHA-1 be phased out as soon as possible.

These attacks are potentially slightly easier when the server provides the Diffie-Hellman parameters such as using the [RFC4419] generated set of diffie-hellman parameters with SHA-1 hashing. If there is a need for using SHA-1 in a Key Exchange for compatibility, it would be desirable it be listed last in the preference list of key exchanges.

Use of the SHA-2 family of hashes found in [RFC6234] rather than the SHA-1 hash is strongly advised.

When it comes to the SHA-2 family of Secure Hashing functions, SHA2-224 has 112 bits of security strength; SHA2-256 has 128 bits of security strength; SHA2-384 has 192 bits of security strength; and SHA2-512 has 256 bits of security strength. As the same compute power is needed for both SHA2-224 and SHA2-256 and currently no KeX uses SHA2-224, it is suggested that the minimum secure hashing function that should be used for Key Exchange Methods is SHA2-256.

To avoid combinatorial explosion of key exchange names, newer key exchanges are restricting to the use of *-sha256 and *-sha512.

1.2. Selecting an appropriate Public Key Algorithm

SSH uses mathematically hard problems for doing Key Exchange:

* Elliptic Curve Cryptography (ECC) has families of curves for Key Exchange Methods for SSH. NIST prime curves with names and other curves are available using an object identifier (OID) with Elliptic Curve Diffie-Hellman (ECDH) via [RFC5656]. Curve25519 and Curve448 key exchanges are used with ECDH via [RFC8731].

* Finite Field Cryptography (FFC) is used for Diffie-Hellman (DH) key exchange with "safe primes" either from a specified list found in [RFC3526] or generated dynamically via [RFC4419] as updated by [RFC8270].

* Integer Factorization Cryptography (IFC) using the RSA algorithm is provided for in [RFC4432].

It is desirable for the security strength of the key exchange be chosen to be comparable with the security strength of the other elements of the SSH handshake. Attackers can target the weakest element of the SSH handshake.

It is desirable to select a minimum of 112 bits of security strength. Based on implementer security needs, a stronger minimum may be desired.

1.2.1. Elliptic Curve Cryptography (ECC)

For ECC, it is recommended to select one with approximately 128 bits of security strength.

Curve Name	Estimated Security Strength
nistp256	128 bits
nistp384	192 bits
nistp521	512 bits
Curve25519	128 bits
Curve448	224 bits

Table 1: ECC Security Strengths

1.2.2. Finite Field Cryptography (FFC)

For FFC, a modulus 2048 bits (112 bits of security strength).

Prime Field Size	Estimated Security Strength	Example MODP Group
2048-bit	112 bits	group14
3072-bit	128 bits	group15

Prime Field Size	Estimated Security Strength	Example MODP Group
4096-bit	152 bits	group16
6144-bit	176 bits	group17
8192-bit	200 bits	group18

Table 2: FFC MODP Security Strengths

The minimum MODP group that MAY be used is the 2048-bit MODP group14. Implementations SHOULD support a 3072-bit MODP group or larger.

1.2.3. Integer Factorization Cryptography (IFC)

The only IFC algorithm for key exchange is the RSA algorithm via [RFC4432]. The minimum modulus size is 2048 bits. The use of a SHA-2 Family hash with RSA 2048-bit keys has sufficient security. The rsa1024-sha1 key exchange has less than 2048 bits and MUST NOT be implemented.

Key Exchange Method	Estimated Security Strength
rsa1024-sha1	80 bits
rsa2048-sha256	112 bits

Table 3: IFC Security Strengths

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Key Exchange Methods

This memo adopts the style and conventions of [RFC4253] in specifying how the use of data key exchange is indicated in SSH.

This RFC also collects key exchange method names in various existing RFCs [RFC4253], [RFC4419], [RFC4432], [RFC4462], [RFC5656], [RFC8268], [RFC8731], [RFC8732], and [RFC8308], and provides a suggested suitability for implementation of MUST, SHOULD, SHOULD NOT, and MUST NOT. Any method not explicitly listed MAY be implemented.

This document is intended to provide guidance as to what key exchange algorithms are to be considered for new or updated SSH implementations.

3.1. SHA-1 and SHA-2 Hashing

All of the key exchanges using the SHA-1 hashing algorithm should be deprecated and phased out of use because SHA-1 has security concerns provided in [RFC6194]. The SHA-2 Family of hashes [RFC6234] is the only one which is more secure than SHA-1 and has been standardized for use with SSH key exchanges.

diffie-hellman-group1-sha1 and diffie-hellman-group14-sha1 are currently mandatory to implement (MTI). diffie-hellman-group14-sha1 is the stronger of the two. Group14 (a 2048-bit MODP group) is defined in [RFC3526]. It is reasonable to retain the diffie-hellman-group14-sha1 exchange for interoperability with legacy implementations. The diffie-hellman-group14-sha1 key exchange MAY be implemented.

The diffie-hellman-group1-sha1, diffie-hellman-group-exchange-sha1, gss-gex-sha1-*, and gss-group1-sha1-* key exchanges SHOULD NOT be implemented.

3.2. Elliptic Curve Cryptography (ECC)

3.2.1. curve25519-sha256 and gss-curve25519-sha256-*

Curve25519 is efficient on a wide range of architectures with properties that allow higher performance implementations compared to traditional elliptic curves. The use of SHA2-256 (also known as SHA-256 and sha256) as defined in [RFC6234] for integrity is a reasonable one for this method. These key exchange methods are described in [RFC8731] and [RFC8732] and is similar to the IKEv2 Key Agreement described in [RFC8031]. The curve25519-sha256 key exchange method has multiple implementations and SHOULD be implemented. The gss-curve25519-sha256-* key exchange method SHOULD also be implemented because it shares the same performance and security characteristics as curve25519-sha2.

3.2.2. curve448-sha512 and gss-curve448-sha512-*

Curve448 provides more security strength than Curve25519 at a higher computational and bandwidth cost. It uses SHA2-512 (also known as SHA-512) defined in [RFC6234] for integrity. This Key Exchange Method is described in [RFC8731] and is similar to the IKEv2 Key Agreement described in [RFC8031]. This method MAY be implemented. The gss-curve448-sha512-* key exchange method MAY also be implemented because it shares the same performance and security characteristics as curve448-sha512.

3.2.3. ECC diffie-hellman using ecdh-*, ecmqv-sha2, and gss-nistp*

The ecdh-sha2-* name-space allows for other curves to be defined for the elliptic curve Diffie Hellman key exchange. At present, there are three named curves in this name-space which SHOULD be supported. They appear in [RFC5656] in section 10.1 Required Curves all of the NISTP curves named are mandatory to implement if any of this RFC is implemented. This set of methods MAY be implemented. If implemented, the named curves SHOULD always be enabled unless specifically disabled by local security policy. In [RFC5656], section 6.1, the method to name other ECDH curves using OIDs is specified. These other curves MAY be implemented.

The GSS-API name-space with gss-nistp*-sha* mirrors the algorithms used by ecdh-sha2-* names. The table provides guidance for implementation.

ECDH reduces bandwidth of key exchanges compared to FFC DH at a similar security strength.

The following table lists algorithms as SHOULD where implementations may be more efficient or widely deployed. The items listed as MAY are potentially less efficient.

Key Exchange Method Name	Guidance
ecdh-sha2-*	MAY
ecdh-sha2-nistp256	SHOULD
gss-nistp256-sha256-*	SHOULD
ecdh-sha2-nistp384	SHOULD
gss-nistp384-sha384-*	SHOULD
ecdh-sha2-nistp521	SHOULD
gss-nistp521-sha512-*	SHOULD
ecmqv-sha2	MAY

Table 4: ECDH Implementation Guidance

It is advisable to match the ECDSA and ECDH algorithms to use the same curve for both to maintain the same security strength in the connection.

3.3. Finite Field Cryptography (FFC)

3.3.1. FFC diffie-hellman using generated MODP groups

This random selection from a set of pre-generated moduli for key exchange uses SHA2-256 as defined in [RFC4419]. [RFC8270] mandates implementations avoid any MODP group whose modulus size is less than 2048 bits. Care should be taken in the pre-generation of the moduli

P and generator G such that the generator provides a Q-ordered subgroup of P. Otherwise, the parameter set may leak one bit of the shared secret leaving the MODP group half as strong. This key exchange MAY be used.

3.3.2. FFC diffie-hellman using named MODP groups

diffie-hellman-group14-sha256 represents the smallest FFC DH key exchange method considered to be secure. It is a reasonably simple transition from SHA-1 to SHA-2. diffie-hellman-group14-sha256 method MUST be implemented. The rest of the FFC MODP groups MAY be implemented. The table below provides explicit guidance by name.

Key Exchange Method Name	Guidance
diffie-hellman-group14-sha256	MUST
gss-group14-sha256-*	SHOULD
diffie-hellman-group15-sha512	MAY
gss-group15-sha512-*	MAY
diffie-hellman-group16-sha512	SHOULD
gss-group16-sha512-*	MAY
diffie-hellman-group17-sha512	MAY
gss-group17-sha512-*	MAY
diffie-hellman-group18-sha512	MAY
gss-group18-sha512-*	MAY

Table 5: FFC Implementation Guidance

3.4. Integer Factorization Cryptography (IFC)

The rsa2048-sha256 key exchange method is defined in [RFC4432]. Uses an RSA 2048-bit modulus with a SHA2-256 hash. This key exchange meets 112 bit minimum security strength. This method MAY be implemented.

3.5. Secure Shell Extension Negotiation

There are two key exchange methods, ext-info-c and ext-info-s, defined in [RFC8308] which are not actually key exchanges. They provide a method to support other Secure Shell negotiations. Being able to extend functionality is desirable. This method SHOULD be implemented.

4. Summary Guidance for Key Exchange Method Names Implementations

The Implement column is the current recommendations of this RFC. Key Exchange Method Names are listed alphabetically.

Key Exchange Method Name	Reference	Implement
curve25519-sha256	RFC8731	SHOULD
curve448-sha512	RFC8731	MAY
diffie-hellman-group-exchange-sha1	RFC4419	SHOULD NOT
diffie-hellman-group-exchange-sha256	RFC4419	MAY
diffie-hellman-group1-sha1	RFC4253	SHOULD NOT
diffie-hellman-group14-sha1	RFC4253	MAY
diffie-hellman-group14-sha256	RFC8268	MUST
diffie-hellman-group15-sha512	RFC8268	MAY
diffie-hellman-group16-sha512	RFC8268	SHOULD
diffie-hellman-group17-sha512	RFC8268	MAY
diffie-hellman-group18-sha512	RFC8268	MAY
ecdh-sha2-*	RFC5656	MAY
ecdh-sha2-nistp256	RFC5656	SHOULD
ecdh-sha2-nistp384	RFC5656	SHOULD
ecdh-sha2-nistp521	RFC5656	SHOULD
ecmqv-sha2	RFC5656	MAY
ext-info-c	RFC8308	SHOULD
ext-info-s	RFC8308	SHOULD
gss-*	RFC4462	MAY
gss-curve25519-sha256-*	RFC8732	SHOULD
gss-curve448-sha512-*	RFC8732	MAY
gss-gex-sha1-*	RFC4462	SHOULD NOT
gss-group1-sha1-*	RFC4462	SHOULD NOT
gss-group14-sha256-*	RFC8732	SHOULD
gss-group15-sha512-*	RFC8732	MAY
gss-group16-sha512-*	RFC8732	SHOULD
gss-group17-sha512-*	RFC8732	MAY
gss-group18-sha512-*	RFC8732	MAY
gss-nistp256-sha256-*	RFC8732	SHOULD
gss-nistp384-sha384-*	RFC8732	SHOULD
gss-nistp521-sha512-*	RFC8732	SHOULD
rsa1024-sha1	RFC4432	MUST NOT
rsa2048-sha256	RFC4432	MAY

Table 6: IANA guidance for key exchange method name implementations

The full set of official [IANA-KEX] key algorithm method names not otherwise mentioned in this document MAY be implemented.

[TO BE REMOVED: This registration should take place at the following location URL: http://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-16]

5. Acknowledgements

Thanks to the following people for review and comments: Denis Bider, Peter Gutmann, Damien Miller, Niels Moeller, Matt Johnston, Iwamoto Kouichi, Simon Josefsson, Dave Dugal, Daniel Migault, Anna Johnston, Tero Kivinen, and Travis Finkenauer.

Thanks to the following people for code to implement interoperable exchanges using some of these groups as found in an this draft: Darren Tucker for OpenSSH and Matt Johnston for Dropbear. And thanks to Iwamoto Kouichi for information about RLogin, Tera Term (ttssh) and Poderosa implementations also adopting new Diffie-Hellman groups based on this draft.

6. Security Considerations

This SSH protocol provides a secure encrypted channel over an insecure network. It performs server host authentication, key exchange, encryption, and integrity checks. It also derives a unique session ID that may be used by higher-level protocols.

Full security considerations for this protocol are provided in [RFC4251].

It is desirable to deprecate or remove key exchange method name that are considered weak. A key exchange method may be weak because too few bits are used, or the hashing algorithm is considered too weak.

7. IANA Considerations

IANA is requested to annotate entries in [IANA-KEX] which MUST NOT be implemented as being deprecated by this document.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, DOI 10.17487/RFC3526, May 2003, <https://www.rfc-editor.org/info/rfc3526>.

[RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, DOI 10.17487/

RFC4250, January 2006, <https://www.rfc-editor.org/info/rfc4250>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC8031] Nir, Y. and S. Josefsson, "Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement", RFC 8031, DOI 10.17487/RFC8031, December 2016, <https://www.rfc-editor.org/info/rfc8031>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8268] Baushke, M., "More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)", RFC 8268, DOI 10.17487/RFC8268, December 2017, <https://www.rfc-editor.org/info/rfc8268>.

[RFC8270] Velvindron, L. and M. Baushke, "Increase the Secure Shell Minimum Recommended Diffie-Hellman Modulus Size to 2048 Bits", RFC 8270, DOI 10.17487/RFC8270, December 2017, <https://www.rfc-editor.org/info/rfc8270>.

[RFC8308] Bider, D., "Extension Negotiation in the Secure Shell (SSH) Protocol", RFC 8308, DOI 10.17487/RFC8308, March 2018, <https://www.rfc-editor.org/info/rfc8308>.

8.2. Informative References

[IANA-KEX] Internet Assigned Numbers Authority (IANA), "Secure Shell (SSH) Protocol Parameters: Key Exchange Method Names", December 2020, <http://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-16>.

[RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <https://www.rfc-editor.org/info/rfc4251>.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <https://www.rfc-editor.org/info/rfc4419>.

[RFC4432] Harris, B., "RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4432, DOI 10.17487/

RFC4432, March 2006, <https://www.rfc-editor.org/info/rfc4432>.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, DOI 10.17487/RFC4462, May 2006, <https://www.rfc-editor.org/info/rfc4462>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011, <https://www.rfc-editor.org/info/rfc6194>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC8731] Adamantiadis, A., Josefsson, S., and M. Baushke, "Secure Shell (SSH) Key Exchange Method Using Curve25519 and Curve448", RFC 8731, DOI 10.17487/RFC8731, February 2020, <https://www.rfc-editor.org/info/rfc8731>.

[RFC8732] Sorce, S. and H. Kario, "Generic Security Service Application Program Interface (GSS-API) Key Exchange with SHA-2", RFC 8732, DOI 10.17487/RFC8732, February 2020, <https://www.rfc-editor.org/info/rfc8732>.

[TRANS-COLL] Bhargavan, K. and G. Leurent, "Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH", Network and Distributed System Security Symposium - NDSS 2016, Feb 2016, San Diego, United States. 10.14722/ndss. 2016.23418 . hal-01244855, <https://hal.inria.fr/hal-01244855/document>.

Author's Address

Mark D. Baushke
Juniper Networks, Inc.

Email: mdb@juniper.net