Internet Engineering Task Force
Internet-Draft
Updates: 4250 (if approved)
Intended status: Standards Track
Expires: January 14, 2021

M. Baushke
Juniper Networks, Inc.

July 13, 2020

Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH)
draft-ietf-curdle-ssh-kex-sha2-11

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.

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 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].

New key exchange methods will use the SHA-2 family of hashes found in [RFC6234] rather than the SHA-1 hash which is in the process of being deprecated for many purposes as no longer providing enough security.

SSH uses multiple mathematically hard problems for doing Key Exchange. Finite Field Cryptography (FFC) with "safe primes" for diffie-hellman (DH) key exchange. Elliptic Curve Cryptography (ECC) using NIST prime curves with Elliptic Curve Diffie-Hellman (ECDH) and the similar Curve25519 and Curve448 key exchanges.

For FFC, many experts have suggested that a prime field of 2048-bits is the minimum (2048-bits is said to have 112 bits of security and 3072-bits is said to have 128 bits of security) allowed and larger sizes up to 8192 bits are considered to be much stronger. The minimum MODP group that MAY be used is the 2048-bit MODP group14.

For ECC, many experts have suggested that a 256-bits curve is the minimum allowed (256-bits is said to have 128 bits of security) and larger sizes up to 521-bits are considered to be much stronger (521-bits are considered to have around 256-bits of security).

When it comes to Secure Hashing functions, SHA2-256 is said to have 128-bits of security SHA2-384 to have 192-bits of security, and SHA2-512 to have 256-bits of security. The older SHA-1 hash is supposed to have about 80-bits of security. The minimum secure hashing function that should be used is SHA2-256 in the year of this RFC.

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. This document will be superseded when one or more of the listed algorithms are considered too weak to continue to use securely, in which case they will likely be downgraded to one of MAY, SHOULD NOT, or MUST NOT. Or, when newer methods have been analyzed and found to be secure with wide enough adoption, upgrade their recommendation from MAY to SHOULD or MUST.

3.1. 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. This Key Exchange Method is described in [RFC8731] and is similar to the IKEv2 Key Agreement described in [RFC8031]. This Key Exchange Method has multiple implementations and SHOULD be implemented in any SSH implementation interested in using elliptic curve based key exchanges.

3.2. curve448-sha512

The Curve448 requires more work than Curve25519. 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.

3.3. diffie-hellman-group-exchange-sha1

This random selection from a set of pre-generated moduli for key exchange uses SHA-1 as defined in [RFC4419]. However, SHA-1 has security concerns provided in [RFC6194], so it would be better to use

a key exchange method which uses a SHA-2 hash as in [RFC6234] for integrity. This key exchange SHOULD NOT be used.

3.4. diffie-hellman-group-exchange-sha256

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 with 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 or the parameter set may leak one bit of the shared private key leaving the MODP group half as strong as desired as compared with the number of bits. This key exchange MAY be used.

3.5. diffie-hellman-group1-sha1

This method is decribed in [RFC4253] and uses [RFC7296] Oakley Group 2 (a 1024-bit MODP group) and SHA-1 [RFC3174]. Due to recent security concerns with SHA-1 [RFC6194] and with MODP groups with less than 2048 bits (see [LOGJAM] and [NIST-SP-800-131Ar2]), this method is considered insecure. This method is being moved from MUST to SHOULD NOT instead of MUST NOT only to allow a transition time to get off of it. There are many old implementations out there that may still need to use this key exchange; it should be removed from server implementations as quickly as possible.

3.6. diffie-hellman-group14-sha1

This method uses [RFC3526] group14 (a 2048-bit MODP group) which is still a reasonable size. This key exchange group uses SHA-1 which has security concerns [RFC6194]. However, this group is still strong enough and is widely deployed. This method is being moved from MUST to SHOULD to aid in transition to stronger SHA-2 based hashes. This method will transition to SHOULD NOT when SHA-2 alternatives are more generally available.

3.7. diffie-hellman-group14-sha256

This key exchange method is defined in [RFC8268] and uses the group14 (a 2048-bit MODP group) along with a SHA-2 (SHA2-256) hash as in [RFC6234] for integrity. This represents the smallest FFC DH key exchange method considered to be secure. It is a reasonably simple transition to move from SHA-1 to SHA-2. This method SHOULD be implemented.

3.8. diffie-hellman-group15-sha512

This key exchange method is defined in [RFC8268] and uses group15 (a 3072-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. This modulus is the minimum required by [CNSA-SUITE]. This method MAY be implemented.

3.9. diffie-hellman-group16-sha512

This key exchange method is defined in [RFC8268] and uses group16 (a 4096-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. The use of FFC DH is well understood and trusted. Adding larger modulus sizes and protecting with SHA2-512 should give enough head room to be ready for the next scare that someone has pre-computed it. This method MAY be implemented.

3.10. diffie-hellman-group17-sha512

This key exchange method is defined in [RFC8268] and uses group17 (a 6144-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. The use of this 6144-bit MODP group is going to be slower than what may be desirable. It is provided to help those who wish to avoid using ECC algorithms. This method MAY be implemented.

3.11. diffie-hellman-group18-sha512

This key exchange method is defined in [RFC8268] and uses group18 (a 8192-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. The use of this 8192-bit MODP group is going to be slower than what may be desirable. It is provided to help those who wish to avoid using ECC algorithms. This method MAY be implemented.

3.12. ecdh-sha2-*

This namespace allows for other curves to be defined for the elliptic curve Diffie Hellman key exchange. At present, there are three members of this namespace They appear in [RFC5656] which are covered below. This set of methods MAY be implemented.

3.12.1. ecdh-sha2-nistp256

This key exchange method is defined in [RFC5656]. ECDH methods are often implemented because they are smaller and faster than using large FFC DH parameters. However, given [CNSA-SUITE] and [safe-curves], this curve may not be as useful and strong as desired for handling TOP SECRET information for some applications. The SSH

development community is divided on this and many implementations do exist. If traditional ECDH key exchange methods are implemented, then this method SHOULD be implemented.

It is advisable to match the ECDSA and ECDH algorithms to use the same curve for both.

3.12.2. ecdh-sha2-nistp384

This key exchange method is defined in [RFC5656]. This ECDH method should be implemented because it is smaller and faster than using large FFC primes with traditional DH. Given [CNSA-SUITE], it is considered good enough for TOP SECRET. However, given [ECDSA-Nonce-Leak], care must be used when using this algorithm. If traditional ECDH key exchange methods are implemented, then this method SHOULD be implemented.

Concerns raised in [safe-curves] may mean that this algorithm will need to be downgraded in the future along the other ECDSA NIST Prime curves.

3.12.3. ecdh-sha2-nistp521

This key exchange method is defined in [RFC5656]. This ECDH method may be implemented because it is smaller and faster than using large FFC DH parameters. It is not listed in [CNSA-SUITE], so it is not currently appropriate for TOP SECRET. It is possible that the mismatch between the 521-bit key and the 512-bit hash could mean that as many as nine bits of this key could be at risk of leaking if appropriate padding measures are not taken. This method MAY be implemented.

3.13. ecmqv-sha2

This key exchange method is defined in [RFC5656]. This method MAY be implemented.

3.14. ext-info-c

This key exchange method is defined in [RFC8308]. This method is not actually a key exchange, but proivides a method to provide support for extensions to other Secure Shell negotations. Being able to extend functionality is desirable, This method SHOULD be implemented.

3.15. ext-info-s

This key exchange method is defined in [RFC8308]. This method is not actually a key exchange, but proivides a method to provide support for extensions to other Secure Shell negotations. Being able to extend functionality is desirable, This method SHOULD be implemented.

3.16. gss-*

This family of key exchange methods is defined in [RFC4462] and [RFC8732] for the GSS-API key exchange methods. The family of methods MAY be implemented for those who need GSS-API methods.

3.16.1. gss-gex-sha1-*

This key exchange method is defined in [RFC4462]. This set of ephemerally generated key exchange groups uses SHA-1 which has security concerns [RFC6194]. This key exchange SHOULD NOT be used. It is intended that it move to MUST NOT as soon as the majority of server implementations no longer offer it. It should be removed from server implementations as quickly as possible.

3.16.2. gss-group1-sha1-*

This key exchange method is defined in [RFC4462]. This method suffers from the same problems of diffie-hellman-group1-sha1. It uses [RFC7296] Oakley Group 2 (a 1024-bit MODP group) and SHA-1 [RFC3174]. Due to recent security concerns with SHA-1 [RFC6194] and with MODP groups with less than 2048 bits (see [LOGJAM] and [NIST-SP-800-131Ar2]), this method is considered insecure. This method SHOULD NOT be implemented. It is intended that it move to MUST NOT as soon as the majority of server implementations no longer offer it. It should be removed from server implementations as quickly as possible.

3.16.3. gss-group14-sha1-*

This key exchange method is defined in [RFC4462]. This generated key exchange groups uses SHA-1 which has security concerns [RFC6194]. If GSS-API key exchange methods are being used, then this one SHOULD be implemented until such time as SHA-2 variants may be implemented and deployed. This method will transition to SHOULD NOT when SHA-2 alternatives are more generally available. No other standard indicated that this method was anything other than optional even though it was implemented in all GSS-API systems. This method MAY be implemented.

3.16.4. gss-group14-sha256-*

This key exchange method is defined in [RFC8732]. This key exchange uses the group14 (a 2048-bit MODP group) along with a SHA-2 (SHA2-256) hash. This represents the smallest Finite Field Cryptography (FFC) Diffie-Hellman (DH) key exchange method considered to be secure. It is a reasonably simple transition to move from SHA-1 to SHA-2. If the GSS-API is to be used, then this method SHOULD be implemented.

3.16.5. gss-group15-sha512-*

This key exchange method is defined in [RFC8732] and uses group15 (a 3072-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. This modulus is the minimum required by [CNSA-SUITE]. If the GSS-API is to be used, then this method MAY be implemented.

3.16.6. gss-group16-sha512-*

This key exchange method is defined in [RFC8732] and uses group16 (a 4096-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. The use of FFC DH is well understood and trusted. Adding larger modulus sizes and protecting with SHA2-512 should give enough head room to be ready for the next scare that someone has pre-computed it. If the GSS-API is to be used, then this method MAY be implemented.

3.16.7. gss-group17-sha512-*

This key exchange method is defined in [RFC8732]and uses group17 (a 6144-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. The use of this 6144-bit MODP group is going to be slower than what may be desirable. It is provided to help those who wish to avoid using ECC algorithms. If the GSS-API is to be used, then this method MAY be implemented.

3.16.8. gss-group18-sha512-*

This key exchange method is defined in [RFC8732] and uses group18 (a 8192-bit MODP group) along with a SHA-2 (SHA2-512) hash as in [RFC6234] for integrity. The use of this 8192-bit MODP group is going to be slower than what may be desirable. It is provided to help those who wish to avoid using ECC algorithms. If the GSS-API is to be used, then this method MAY be implemented.

3.16.9. gss-nistp256-sha256-*

This key exchange method is defined in [RFC8732]. If the GSS-API is to be used with ECC algorithms, then this method SHOULD be implemented.

3.16.10. gss-nistp384-sha384-*

This key exchange method is defined in [RFC8732]. If the GSS-API is to be used with ECC algorithms, then this method SHOULD be implemented to permit TOP SECRET information to be communicated.

3.16.11. gss-nistp521-sha512-*

This key exchange method is defined in [RFC8732]. If the GSS-API is to be used with ECC algorithms, then this method MAY be implemented.

3.16.12. gss-curve25519-sha256-*

This key exchange method is defined in [RFC8732]. If the GSS-API is to be used with ECC algorithms, then this method SHOULD be implemented.

3.16.13. gss-curve448-sha512-*

This key exchange method is defined in [RFC8732]. If the GSS-API is to be used with ECC algorithms, then this method MAY be implemented.

3.17. rsa1024-sha1

This key exchange method is defined in [RFC4432]. The security of RSA 1024-bit modulus keys is not good enough any longer per [NIST-SP-800-131Ar2], an RSA key size should be a minimum of 2048-bits. This key exchange group uses SHA-1 which has security concerns [RFC6194]. This method MUST NOT be implemented.

3.18. rsa2048-sha256

This key exchange method is defined in [RFC4432]. An RSA 2048-bit modulus key with a SHA2-256 hash. At the present time, a 2048-bit RSA key is considered to be suffiently strong in [NIST-SP-800-131Ar2] to be permitted. In addition, the use of a SHA-2 hash as defined in [RFC6234] is a good integrity measure. This method MAY be implemented.

4. Selecting an appropriate hashing algorithm

As may be seen from the above, the Key Exchange Methods area all using either SHA256 or SHA512 with the exception of the ecdh-sha2-nistp384 which uses SHA384.

The cited CNSA Suite specifies the use of SHA384 and says that SHA256 is no longer good enough for TOP SECRET. Nothing is said about the use of SHA512. It may be that the internal state of 1024 bits in both SHA384 and SHA512 makes the SHA384 more secure because it does not leak an additional 128 bits of state. Of course, the use of SHA384 also reduces the security strength to 384 bits instead of being 512 bits. This seems to contradict the desire to double the symmetric key strength in order to try to be safe from Post Quantum Computing (PQC) attacks given a session key derived from the key exchange will be limited to the security strength of the hash being used.

The move away from SHA256 to SHA512 for the newer key exchange methods is more to try to slow Grover's algorithm (a PQC attack) slightly. It is also the case that SHA2-512 may, in many modern CPUs, be implemented more efficiently using 64-bit arithmetic than SHA256 which is faster on 32-bit CPUs. The selection of SHA384 vs SHA512 is more about reducing the number of code point alternatives to negotiate. There seemed to be consensus in favor of SHA2-512 over SHA2-384 for key exchanges.

5. Summary Guidance for Key Exchange Method Names

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	SHOULD
diffie-hellman-group14-sha256	RFC8268	SHOULD
diffie-hellman-group15-sha512	RFC8268	MAY
diffie-hellman-group16-sha512	RFC8268	MAY
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	MAY
ecmqv-sha2	RFC5656	MAY
ext-info-c	RFC8308	MAY
ext-info-s	RFC8308	MAY
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	MAY
gss-group17-sha512-*	RFC8732	MAY
gss-group18-sha512-*	RFC8732	MAY
gss-nistp256-sha256-*	RFC8732	SHOULD
gss-nistp384-sha384-*	RFC8732	SHOULD
gss-nistp521-sha512-*	RFC8732	MAY
rsa1024-sha1	RFC4432	MUST NOT
rsa2048-sha256	RFC4432	MAY

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

The guidance of this document is that the SHA-1 algorithm hashing SHOULD NOT be used. If it is used in implementations, it should only be provided for backwards compatibility, should not be used in new designs, and should be phased out of existing key exchanges as quickly as possible because of its known weaknesses. Any key exchange using SHA-1 should not be in a default key exchange list if at all possible. If they are needed for backward compatibility, they SHOULD be listed after all of the SHA-2 based key exchanges.

The [RFC4253] MUST diffie-hellman-group14-sha1 method SHOULD be retained for compatibility with older Secure Shell implementations. It is intended that this key exchange method be phased out as soon as possible. It SHOULD be listed after all possible SHA-2 based key exchanges.

It is believed that all current SSH implementations should be able to achieve an implementation of the "diffie-hellman-group14-sha256" method. To that end, this is one method that MUST be implemented.

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

6. 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, and Tero Kivinen.

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.

7. Security Considerations

This SSH protocol provides a secure encrypted channel over an insecure network. It performs server host authentication, key exchange, encryption, and integrity protection. 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.

The diffie-hellman-group1-sha1 is being moved from MUST to MUST NOT. This method used [RFC7296] Oakley Group 2 (a 1024-bit MODP group) and SHA-1 [RFC3174]. Due to recent security concerns with SHA-1 [RFC6194] and with MODP groups with less than 2048 bits [NIST-SP-800-131Ar2], this method is no longer considered secure.

The United States Information Assurance Directorate (IAD) at the National Security Agency (NSA) has published a FAQ [MFQ-U-OO-815099-15] suggesting that the use of Elliptic Curve Diffie-Hellman (ECDH) using the nistp256 curve and SHA-2 based hashes less than SHA2-384 are no longer sufficient for transport of TOP SECRET information. If your systems need to be concerned with TOP SECRET information, then the guidance for supporting lesser security strength key exchanges may be omitted for your implementations.

The MODP group14 is already required for SSH implementations and most implementations already have a SHA2-256 implementation, so diffie-hellman-group14-sha256 is provided as an easy to implement and faster to use key exchange. Small embedded applications may find this KEX desirable to use.

The NSA Information Assurance Directorate (IAD) has also published the Commercial National Security Algorithm Suite (CNSA Suite) [CNSA-SUITE] in which the 3072-bit MODP Group 15 in [RFC3526] is explicitly mentioned as the minimum modulus to protect TOP SECRET communications.

It has been observed in [safe-curves] that the NIST Elliptic Curve Prime Curves (P-256, P-384, and P-521) are perhaps not the best available for Elliptic Curve Cryptography (ECC) Security. For this reason, none of the [RFC5656] curves are mandatory to implement. However, the requirement that "every compliant SSH ECC implementation MUST implement ECDH key exchange" is now taken to mean that if ecdsa-sha2-[identifier] is implemented, then ecdh-sha2-[identifier] MUST be implemented.

In a Post-Quantum Computing (PQC) world, it will be desirable to use larger cyclic subgroups. To do this using Elliptic Curve Cryptography will require much larger prime base fields, greatly reducing their efficiency. Finite Field based Cryptography already requires large enough base fields to accommodate larger cyclic subgroups. Until such time as a PQC method of key exchange is developed and adopted, it may be desirable to generate new and larger DH groups to avoid pre-calculation attacks that are provably not backdoored.

8. IANA Considerations

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

9. References

9.1. 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>.

9.2. Informative References

[CNSA-SUITE]
NSA/IAD, "Commercial National Security Algorithm Suite", September 2016, <https://apps.nsa.gov/iaarchive/programs/iad-initiatives/cnsa-suite.cfm>.

[ECDSA-Nonce-Leak]
Mulder, E. D., Hutter, M., Marson, M. E., and P. Pearson, "Using Bleichenbacher's Solution to the Hidden Number Problem to Attack Nonce Leaks in 384-Bit ECDSA", IACR Cryptology ePrint Archive 2013, August 2013, <https://eprint.iacr.org/2013/346.pdf>.

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

[LOGJAM] Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J., Heninger, N., Springall, D., Thome, E., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Beguelin, S., and P. Zimmermann, "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice", ACM Conference on Computer and Communications Security (CCS) 2015, 2015, <https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf>.

[MFQ-U-OO-815099-15]
NSA/CSS, "CNSA Suite and Quantum Computing FAQ", January 2016, <https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm>.

[NIST-SP-800-131Ar2]
Barker, E. and A. Roginsky, "Transitions: Recommendation for the Transitioning of the Use of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131A Revision 2, March 2019, <http://doi.org/10.6028/NIST.SP.800-131Ar2.pdf>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[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>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[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>.

[safe-curves]
Bernstein, D. J. and T. Lange, "SafeCurves: choosing safe curves for elliptic-curve cryptography.", February 2016, <https://safecurves.cr.yp.to/>.

Author's Address

Mark D. Baushke
Juniper Networks, Inc.

Email: mdb@juniper.net