Internet Engineering Task Force
Internet-Draft
Updates: 4250 (if approved)
Intended status: Standards Track
Expires: October 17, 2017

M. Baushke
Juniper Networks, Inc.
April 15, 2017

Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH)
draft-ietf-curdle-ssh-kex-sha2-08

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. This document updates [RFC4250].

This document adds recommendations for adoption of Key Exchange Methods which MUST, SHOULD+, SHOULD, SHOULD-, MAY, SHOULD NOT, and MUST NOT be implemented. New key exchange methods will use the SHA-2 family of hashes and are drawn from these ssh-curves from [I-D.ietf-curdle-ssh-curves] and new-modp from the [I-D.ietf-curdle-ssh-modp-dh-sha2] and gss-keyex [NEWGSSAPI].

[TO BE REMOVED: Please send comments on this draft to curdle@ietf.org.]

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

When used in the tables in this document, these terms indicate that the listed algorithm MUST, MUST NOT, SHOULD, SHOULD NOT or MAY be implemented as part of a Secure Shell implementation. Additional terms used in this document are:

SHOULD+ This term means the same as SHOULD. However, it is likely that an algorithm marked as SHOULD+ will be promoted at some future time to be a MUST.

SHOULD- This term means the same as SHOULD. However, an algorithm marked as SHOULD- may be deprecated to a MAY in a future version of this document.

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], [I-D.ietf-curdle-ssh-modp-dh-sha2], [NEWGSSAPI], and [I-D.ietf-curdle-ssh-curves] and provides a suggested suitability for implementation of MUST, SHOULD+, SHOULD, 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, or when newer methods have been analyzed and found to be secure with wide enough adoption to upgrade their recommendation from MAY to SHOULD or MUST.

3.1. curve25519-sha256

The Curve25519 provides strong security and is efficient on a wide range of architectures with properties that allow better implementation properties compared to traditional elliptic curves. The use of SHA2-256 for integrity is a reasonable one for this method. This Key Exchange Method has multiple implementations and SHOULD+ be implemented in any SSH interested in using elliptic curve based key exchanges.

3.2. diffie-hellman-group-exchange-sha1

This set of ephemerally generated key exchange groups uses SHA-1 as defined in [RFC4419]. However, SHA-1 has security concerns provided in [RFC6194]. It is recommended that these key exchange groups NOT be used. This key exchange MUST NOT be used.

3.3. diffie-hellman-group-exchange-sha256

This set of ephemerally generated key exchange groups uses SHA2-256 as defined in [RFC4419]. It is recommended implementations avoid any MODP group with less than 2048 bits. This key exchange MAY be used.

3.4. diffie-hellman-group1-sha1

This method 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 [NIST-SP-800-131Ar1], this method is considered insecure. This method is being moved from MUST to MUST NOT.

3.5. diffie-hellman-group14-sha1

This method uses [RFC3526] group14 (a 2048-bit MODP group) which has no concerns. This generated 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 MUST NOT when SHA-2 alternatives are more generally available.

3.6. diffie-hellman-group14-sha256

This generated key exchange uses a 2048-bit sized 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. This method MUST be implemented.

3.7. diffie-hellman-group16-sha512

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. This modulus is larger than that required by [CNSA-SUITE] and should be sufficient to inter-operate with more paranoid nation-states. This method SHOULD+ be implemented.

3.8. ecdh-sha2-nistp256

Elliptic Curve Diffie-Hellman (ECDH) are often implemented because they are smaller and faster than using large FFC primes with traditional Diffie-Hellman (DH). However, given [CNSA-SUITE] and [safe-curves], this curve may not be as useful and strong as desired. The SSH development community is divided on this and many implementations do exist. However, there are good implementations of this along with a constant-time SHA2-256 implementation. If an implementer does not have a constant-time SHA2-384 implementation (which helps avoid side-channel attacks), then this is the correct ECDH to implement. If traditional ECDH key exchange methods are implemented, then this method SHOULD- be implemented.

3.9. ecdh-sha2-nistp384

This ECDH method should be implemented because it is smaller and faster than using large FFC primes with traditional Diffie-Hellman (DH). Given [CNSA-SUITE], it is considered good enough for TOP SECRET for now. This really needs a constant-time implementation of SHA2-384 to be useful. If traditional ECDH key exchange methods are implemented, then this method SHOULD+ be implemented.

3.10. gss-gex-sha1-*

This set of ephemerally generated key exchange groups uses SHA-1 which has security concerns [RFC6194]. It is recommended that these key exchange groups NOT be used. This key exchange MUST NOT be implemented.

3.11. gss-group1-sha1-*

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 [NIST-SP-800-131Ar1], this method is considered insecure. This method MUST NOT be implemented.

3.12. gss-group14-sha1-*

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.

3.13. gss-group14-sha256-*

If the GSS-API is to be used, then this method SHOULD be implemented.

3.14. gss-group16-sha512-*

If the GSS-API is to be used, then this method SHOULD+ be implemented.

3.15. rsa1024-sha1

The security of RSA 1024-bit modulus keys is not good enough any longer. A minimum bit size should be 2048-bit groups. This generated key exchange groups uses SHA-1 which has security concerns [RFC6194]. This method MUST NOT be implemented.

4. 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	ssh-curves	SHOULD+
diffie-hellman-group-exchange-sha1	RFC4419	MUST NOT
diffie-hellman-group1-sha1	RFC4253	MUST NOT
diffie-hellman-group14-sha1	RFC4253	SHOULD-
diffie-hellman-group14-sha256	new-modp	MUST
diffie-hellman-group16-sha512	new-modp	SHOULD+
ecdh-sha2-nistp256	RFC5656	SHOULD-
ecdh-sha2-nistp384	RFC5656	SHOULD+
gss-gex-sha1-*	RFC4462	MUST NOT
gss-group1-sha1-*	RFC4462	MUST NOT
gss-group14-sha1-*	RFC4462	SHOULD-
gss-group14-sha256-*	gss-keyex	SHOULD
gss-group16-sha512-*	gss-keyex	SHOULD+
rsa1024-sha1	RFC4432	MUST NOT

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 MUST 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>]

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.

Thanks to the following people for code to implement inter-operable 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 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-800-131Ar1], 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. It is for this reason that this draft moves ecdh-sha2-nistp256 from a MUST to MAY as a key exchange method. This is the same reason that the stronger MODP groups being adopted. As the MODP group14 is already present in most 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 precalcualtion attacks that are provably not backdoored.

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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc4253>.

8.2. Informative References

[CNSA-SUITE]
"Information Assurance by the National Security Agency", "Commercial National Security Algorithm Suite", September 2016, <https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm>.

[I-D.ietf-curdle-ssh-curves]
Adamantiadis, A., Josefsson, S., and M. Baushke, "Secure Shell (SSH) Key Exchange Method using Curve25519 and Curve448", draft-ietf-curdle-ssh-curves-04 (work in progress), April 2017.

[I-D.ietf-curdle-ssh-modp-dh-sha2]
Baushke, M., "More Modular Exponential (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)", draft-ietf-curdle-ssh-modp-dh-sha2-04 (work in progress), April 2017.

[IANA-KEX]
Internet Assigned Numbers Authority (IANA), "Secure Shell (SSH) Protocol Parameters: Key Exchange Method Names", March 2017, <http://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-16>.

[MFQ-U-OO-815099-15]
"National Security Agency/Central Security Service", "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>.

[NEWGSSAPI]
Sorce, S. and H. Kario, "GSS-API Key Exchange with SHA2", December 2016, <https://tools.ietf.org/html/draft-ssorce-gss-keyex-sha2-00>.

[NIST-SP-800-131Ar1]
Barker, and Roginsky, "Transitions: Recommendation for the Transitioning of the Use of Cryptographic Algorithms and Key Lengths", NIST Special Publication 800-131A Revision 1, November 2015, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <http://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, <http://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, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc6194>.

[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, <http://www.rfc-editor.org/info/rfc7296>.

[safe-curves]
Bernstein, and 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.
1133 Innovation Way
Sunnyvale, CA 94089-1228
US

Email: mdb@juniper.net
URI: http://www.juniper.net/