Independent Submission
Request for Comments: 9212
Category: Informational
ISSN: 2070-1721

N. Gajcowski
M. Jenkins
NSA

March 2022

Commercial National Security Algorithm (CNSA) Suite Cryptography for Secure Shell (SSH)

Abstract

The United States Government has published the National Security Agency (NSA) Commercial National Security Algorithm (CNSA) Suite, which defines cryptographic algorithm policy for national security applications. This document specifies the conventions for using the United States National Security Agency's CNSA Suite algorithms with the Secure Shell Transport Layer Protocol and the Secure Shell Authentication Protocol. It applies to the capabilities, configuration, and operation of all components of US National Security Systems (described in NIST Special Publication 800-59) that employ Secure Shell (SSH). This document is also appropriate for all other US Government systems that process high-value information. It is made publicly available for use by developers and operators of these and any other system deployments.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9212.

Table of Contents

1. Introduction

This document specifies conventions for using the United States National Security Agency's CNSA Suite algorithms [CNSA] with the Secure Shell Transport Layer Protocol [RFC4253] and the Secure Shell Authentication Protocol [RFC4252]. It applies to the capabilities, configuration, and operation of all components of US National Security Systems (described in NIST Special Publication 800-59 [SP80059]) that employ SSH. This document is also appropriate for all other US Government systems that process high-value information. It is made publicly available for use by developers and operators of these and any other system deployments.

2. Terminology

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. The Commercial National Security Algorithm Suite

The NSA profiles commercial cryptographic algorithms and protocols as part of its mission to support secure, interoperable communications for US Government National Security Systems. To this end, it publishes guidance both to assist with the US Government's transition to new algorithms and to provide vendors -- and the Internet community in general -- with information concerning their proper use and configuration.

Recently, cryptographic transition plans have become overshadowed by the prospect of the development of a cryptographically relevant quantum computer. The NSA has established the Commercial National Security Algorithm (CNSA) Suite to provide vendors and IT users near-term flexibility in meeting their information assurance interoperability requirements using current cryptography. The purpose behind this flexibility is to avoid vendors and customers making two major transitions (i.e., to elliptic curve cryptography and then to post-quantum cryptography) in a relatively short timeframe, as we anticipate a need to shift to quantum-resistant cryptography in the near future.

The NSA is authoring a set of RFCs, including this one, to provide updated guidance concerning the use of certain commonly available commercial algorithms in IETF protocols. These RFCs can be used in conjunction with other RFCs and cryptographic guidance (e.g., NIST Special Publications) to properly protect Internet traffic and data-at-rest for US Government National Security Systems.

4. CNSA and Secure Shell

Several RFCs have documented how each of the CNSA components are to be integrated into Secure Shell (SSH):

kex algorithms:

* ecdh-sha2-nistp384 [RFC5656]

* diffie-hellman-group15-sha512 [RFC8268]

* diffie-hellman-group16-sha512 [RFC8268]

public key algorithms:

* ecdsa-sha2-nistp384 [RFC5656]

* rsa-sha2-512 [RFC8332]

encryption algorithms (both client_to_server and server_to_client):

* AEAD_AES_256_GCM [RFC5647]

message authentication code (MAC) algorithms (both client_to_server and server_to_client):

* AEAD_AES_256_GCM [RFC5647]

While the approved CNSA hash function for all purposes is SHA-384, as defined in [FIPS180], commercial products are more likely to incorporate the kex algorithms and public key algorithms based on SHA-512 (sha2-512), which are defined in [RFC8268] and [RFC8332]. Therefore, the SHA-384-based kex and public key algorithms SHOULD be used; SHA-512-based algorithms MAY be used. Any hash algorithm other than SHA-384 or SHA-512 MUST NOT be used.

Use of the Advanced Encryption Standard in Galois/Counter Mode (AES-GCM) shall meet the requirements set forth in [SP800-38D], with the additional requirements that all 16 octets of the authentication tag MUST be used as the SSH data integrity value and that AES is used with a 256-bit key. Use of AES-GCM in SSH should be done as described in [RFC5647], with the exception that AES-GCM need not be listed as the MAC algorithm when its use is implicit (such as done in aes256-gcm@openssh.com). In addition, [RFC5647] fails to specify that the AES-GCM invocation counter is incremented mod 2^64. CNSA implementations MUST ensure the counter never repeats and is properly incremented after processing a binary packet:

   invocation_counter = invocation_counter + 1 mod 2^64.

The purpose of this document is to draw upon all of these documents to provide guidance for CNSA-compliant implementations of Secure Shell. Algorithms specified in this document may be different from mandatory-to-implement algorithms; where this occurs, the latter will be present but not used. Note that, while compliant Secure Shell implementations MUST follow the guidance in this document, that requirement does not in and of itself imply that a given implementation of Secure Shell is suitable for use national security systems. An implementation must be validated by the appropriate authority before such usage is permitted.

5. Security Mechanism Negotiation and Initialization

As described in Section 7.1 of [RFC4253], the exchange of SSH_MSG_KEXINIT between the server and the client establishes which key agreement algorithm, MAC algorithm, host key algorithm (server authentication algorithm), and encryption algorithm are to be used.

This section specifies the use of CNSA components in the Secure Shell algorithm negotiation, key agreement, server authentication, and user authentication.

The choice of all but the user authentication methods is determined by the exchange of SSH_MSG_KEXINIT between the client and the server.

The kex_algorithms name-list is used to negotiate a single key agreement algorithm between the server and client in accordance with the guidance given in Section 4. While [RFC9142] establishes general guidance on the capabilities of SSH implementations and requires support for "diffie-hellman-group14-sha256", it MUST NOT be used. The result MUST be one of the following kex_algorithms, or the connection MUST be terminated:

* ecdh-sha2-nistp384 [RFC5656]

* diffie-hellman-group15-sha512 [RFC8268]

* diffie-hellman-group16-sha512 [RFC8268]

One of the following sets MUST be used for the encryption_algorithms and mac_algorithms name-lists. Either set MAY be used for each direction (i.e., client_to_server or server_to_client), but the result must be the same (i.e., use of AEAD_AES_256_GCM).

```
encryption_algorithm_name_list := { AEAD_AES_256_GCM }

mac_algorithm_name_list := { AEAD_AES_256_GCM }
```

or

```
encryption_algorithm_name_list := { aes256-gcm@openssh.com }

mac_algorithm_name_list := {}
```

One of the following public key algorithms MUST be used:

* rsa-sha2-512 [RFC8332]

* ecdsa-sha2-nistp384 [RFC5656]

The procedures for applying the negotiated algorithms are given in the following sections.

6. Key Exchange

The key exchange to be used is determined by the name-lists exchanged in the SSH_MSG_KEXINIT packets, as described above. Either Elliptic Curve Diffie-Hellman (ECDH) or Diffie-Hellman (DH) MUST be used to establish a shared secret value between the client and the server.

A compliant system MUST NOT allow the reuse of ephemeral/exchange values in a key exchange algorithm due to security concerns related to this practice. Section 5.6.3.3 of [SP80056A] states that an ephemeral private key shall be used in exactly one key establishment transaction and shall be destroyed (zeroized) as soon as possible. Section 5.8 of [SP80056A] states that such shared secrets shall be destroyed (zeroized) immediately after its use. CNSA-compliant systems MUST follow these mandates.

6.1. ECDH Key Exchange

The key exchange begins with the SSH_MSG_KEXECDH_INIT message that contains the client’s ephemeral public key used to generate a shared secret value.

The server responds to an SSH_MSG_KEXECDH_INIT message with an SSH_MSG_KEXECDH_REPLY message that contains the server's ephemeral public key, the server's public host key, and a signature of the exchange hash value formed from the newly established shared secret value. The kex algorithm MUST be ecdh-sha2-nistp384, and the public key algorithm MUST be either ecdsa-sha2-nistp384 or rsa-sha2-512.

6.2. DH Key Exchange

The key exchange begins with the SSH_MSG_KEXDH_INIT message that contains the client's DH exchange value used to generate a shared secret value.

The server responds to an SSH_MSG_KEXDH_INIT message with an SSH_MSG_KEXDH_REPLY message that contains the server's DH exchange value, the server's public host key, and a signature of the exchange hash value formed from the newly established shared secret value. The kex algorithm MUST be one of diffie-hellman-group15-sha512 or diffie-hellman-group16-sha512, and the public key algorithm MUST be either ecdsa-sha2-nistp384 or rsa-sha2-512.

7. Authentication

7.1. Server Authentication

A signature on the exchange hash value derived from the newly established shared secret value is used to authenticate the server to the client. Servers MUST be authenticated using digital signatures. The public key algorithm implemented MUST be ecdsa-sha2-nistp384 or rsa-sha2-512. The RSA public key modulus MUST be 3072 or 4096 bits in size; clients MUST NOT accept RSA signatures from a public key modulus of any other size.

The following public key algorithms MUST be used:

* ecdsa-sha2-nistp384 [RFC5656]

* rsa-sha2-512 [RFC8332]

The client MUST verify that the presented key is a valid authenticator for the server before verifying the server signature. If possible, validation SHOULD be done using certificates. Otherwise, the client MUST validate the presented public key through some other secure, possibly off-line mechanism. Implementations MUST NOT employ a "Trust on First Use (TOFU)" security model where a client accepts the first public host key presented to it from a not-yet-verified server. Use of a TOFU model would allow an intermediate adversary to present itself to the client as the server.

Where X.509 v3 Certificates are used, their use MUST comply with [RFC8603].

7.2. User Authentication

The Secure Shell Transport Layer Protocol authenticates the server to the host but does not authenticate the user (or the user's host) to the server. All users MUST be authenticated, MUST follow [RFC4252], and SHOULD be authenticated using a public key method. Users MAY authenticate using passwords. Other methods of authentication MUST not be used, including "none".

When authenticating with public key, the following public key algorithms MUST be used:

* ecdsa-sha2-nistp384 [RFC5656]

* rsa-sha2-512 [RFC8332]

The server MUST verify that the public key is a valid authenticator for the user. If possible, validation SHOULD be done using certificates. Otherwise, the server must validate the public key through another secure, possibly off-line mechanism.

Where X.509 v3 Certificates are used, their use MUST comply with [RFC8603].

If authenticating with RSA, the client's public key modulus MUST be 3072 or 4096 bits in size, and the server MUST NOT accept signatures from an RSA public key modulus of any other size.

To facilitate client authentication with RSA using SHA-512, clients and servers SHOULD implement the server-sig-algs extension, as specified in [RFC8308]. In that case, in the SSH_MSG_KEXINIT, the client SHALL include the indicator ext-info-c to the kex_algorithms field, and the server SHOULD respond with an SSH_MSG_EXT_INFO message containing the server-sig-algs extension. The server MUST list only ecdsa-sha2-nistp384 and/or rsa-sha2-512 as the acceptable public key algorithms in this response.

If authenticating by passwords, it is essential that passwords have sufficient entropy to protect against dictionary attacks. During authentication, the password MUST be protected in the established encrypted communications channel. Additional guidelines are provided in [SP80063].

8. Confidentiality and Data Integrity of SSH Binary Packets

Secure Shell transfers data between the client and the server using its own binary packet structure. The SSH binary packet structure is independent of any packet structure on the underlying data channel. The contents of each binary packet and portions of the header are encrypted, and each packet is authenticated with its own message authentication code. Use of AES-GCM will both encrypt the packet and form a 16-octet authentication tag to ensure data integrity.

8.1. Galois/Counter Mode

Use of AES-GCM in Secure Shell is described in [RFC5647]. CNSA-complaint SSH implementations MUST support AES-GCM (negotiated as AEAD_AES_GCM_256 or aes256-gcm@openssh; see Section 5) to provide confidentiality and ensure data integrity. No other confidentiality or data integrity algorithms are permitted.

The AES-GCM invocation counter is incremented mod 2^64. That is, after processing a binary packet:

```
invocation_counter = invocation_counter + 1 mod 2^64
```

The invocation counter MUST NOT repeat a counter value.

8.2. Data Integrity

As specified in [RFC5647], all 16 octets of the authentication tag MUST be used as the SSH data integrity value of the SSH binary packet.

9. Rekeying

Section 9 of [RFC4253] allows either the server or the client to initiate a "key re-exchange ... by sending an SSH_MSG_KEXINIT packet" and to "change some or all of the [cipher] algorithms during the re-

exchange". This specification requires the same cipher suite to be employed when rekeying; that is, the cipher algorithms MUST NOT be changed when a rekey occurs.

10. Security Considerations

The security considerations of [RFC4251], [RFC4252], [RFC4253], [RFC5647], and [RFC5656] apply.

11. IANA Considerations

This document has no IANA actions.

12. References

12.1. Normative References

[CNSA] Committee for National Security Systems, "Use of Public Standards for Secure Information Sharing", CNSSP 15, October 2016, <https://www.cnss.gov/CNSS/Issuances/Policies.cfm>.

[FIPS180] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

[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>.

[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>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[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>.

[RFC5647] Igoe, K. and J. Solinas, "AES Galois Counter Mode for the Secure Shell Transport Layer Protocol", RFC 5647, DOI 10.17487/RFC5647, August 2009, <https://www.rfc-editor.org/info/rfc5647>.

[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>.

[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>.

[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>.

[RFC8332] Bider, D., "Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol", RFC 8332, DOI 10.17487/RFC8332, March 2018, <https://www.rfc-editor.org/info/rfc8332>.

[RFC8603] Jenkins, M. and L. Zieglar, "Commercial National Security Algorithm (CNSA) Suite Certificate and Certificate Revocation List (CRL) Profile", RFC 8603, DOI 10.17487/RFC8603, May 2019, <https://www.rfc-editor.org/info/rfc8603>.

12.2. Informative References

[RFC9142] Baushke, M., "Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH)", RFC 9142, DOI 10.17487/RFC9142, January 2022, <https://www.rfc-editor.org/info/rfc9142>.

[SP800-38D] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, DOI 10.6028/NIST.SP.800-38D, November 2007, <https://doi.org/10.6028/NIST.SP.800-38D>.

[SP80056A] National Institute of Standards and Technology, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", Revision 3, NIST Special Publication 800-56A, DOI 10.6028/NIST.SP.800-56Ar3, April 2018, <https://doi.org/10.6028/NIST.SP.800-56Ar3>.

[SP80059] National Institute of Standards and Technology, "Guideline for Identifying an Information System as a National Security System", NIST Special Publication 800-59, DOI 10.6028/NIST.SP.800-59, August 2003, <https://doi.org/10.6028/NIST.SP.800-59>.

[SP80063] National Institute of Standards and Technology, "Digital Identity Guidelines", NIST Special Publication 800-63-3, DOI 10.6028/NIST.SP.800-63-3, June 2017, <https://doi.org/10.6028/NIST.SP.800-63-3>.

Authors' Addresses

Nicholas Gajcowski
National Security Agency
Email: nhgajco@uwe.nsa.gov

Michael Jenkins
National Security Agency
Email: mjjenki@cyber.nsa.gov