Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 19 September 2025

D. Miller
OpenSSH

18 March 2025

Fixed AES-GCM modes for the SSH protocol
draft-miller-sshm-aes-gcm-00

Abstract

This document describes the use of the AES-GCM AEAD in the Secure Shell (SSH) protocol, using the underlying construction of [RFC5647] but fixing problems in the negotiation mechanism.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 19 September 2025.

Table of Contents

1. Introduction

Secure Shell (SSH) is a cryptographic protocol for secure remote connections and login over untrusted networks. The SSH transport layer [RFC4253] uses symmetric encryption to provide a confidential and integrity-protected channel over which application traffic is carried. When initially designed, the SSH protocol negotiated ciphers and MACs separately and combines them using a specified encrypt-and-MAC construction.

[RFC5647] introduced the first AEAD (Authenticated Encryption with Additional Data) in the form of AES-GCM (Galois Counter Mode). This document specified two algorithms: AEAD_AES_128_GCM and AEAD_AES_256_GCM.

Unfortunately, this document contained a problem in how these new modes were negotiated in key exchange: Section 5.1 of [RFC5647] required that these new algorithm identifiers be included in both the "encryption_algorithms" and "mac_algorithms" fields on the SSH_MSG_KEXINIT negotiation message (Section 7.1 of [RFC4253]), and that algorithm negotiation must fail if the endpoints do not agree on the same algorithm for both cipher and MAC.

This aspect of the design is problematic, as it creates the situation where two endpoints could propose well-formed algorithm proposals that contain shared algorithms that should (per [RFC4253]) yield a successful negotiated set of parameters, but could potentially fail to if the endpoints contain the AEAD_AES_128/256_GCM algorithms in different preference positions. As such, these modes were not safe to use in the presence of other cipher/MAC preferences.

Most SSH implementations have since adopted a different approach to negotiating AEAD modes in the key exchange phase. Instead of requiring the same algorithm name to appear in both the "encryption_algorithms" and "mac_algorithms" SSH_MSG_KEXINIT fields, the AEAD appears only in the "encryption_algorithms" field and, if negotiated as the preferred cipher, negotiation of "mac_algorithms" is skipped entriely. This approach was used by [I-D.ietf-sshm-chacha20-poly1305] and by the OpenSSH AES-GCM modes that are being specified here.

1.1. 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.

2. Revised AES-GCM modes

This document defines two new AEAD modes for the SSH transport protocol, "aes128-gcm" and "aes256-gcm". These modes have an identical construction to the AEAD_AES_128_GCM and AEAD_AES_256_GCM (respecitvely) from [RFC5647] except for their negotiation.

Specifically, these modes inherit all provisions of [RFC5647] except those relating to Key Exchange (Section 5.1 of [RFC5647]. Instead, the following algorithm negotiation rules apply:

AEAD modes, such as "aes128-gcm" or "aes256-gcm", MUST be listed in the "encryption_algorithms" field of SSH_MSG_KEXINIT packets only. They MUST NOT appear in the "mac_algorithms" field.

When an AEAD mode is selected as the highest preference of the "encryption_algorithms" by the endpoints, MAC negotiation MUST be skipped and the contents of the "mac_algorithms" field ignored. Specifically, failures of the endpoints to agree on a mutual selection from the "mac_algorithms" field MUST NOT cause connection failure.

This supports the situation where a client and server agree on an AEAD (which provides all the properties that SSH requires of a MAC), but have disjoint sets of proposed MAC algorithms.

2.1. aes128-gcm

This algorithm is identical to AEAD_AES_128_GCM defined in Section 6.1 of [RFC5647] except for the algorithm negotiation changes mentioned above.

2.2. aes256-gcm

This algorithm is identical to AEAD_AES_256_GCM defined in Section 6.2 of [RFC5647] except for the algorithm negotiation changes mentioned above.

3. IANA Considerations

This protocol requires one existing registry to be modified.

3.1. Additions to SSH Encryption Algorithm Names

IANA is requested to insert the following entries into the table Encryption Algorithm Names [IANA-SSH-EXT] under Secure Shell (SSH) Protocol Parameters [RFC4250].

Algorithm name	Reference
aes128-gcm	Section 2.1
aes256-gcm	Section 2.2

Table 1

4. Security Considerations

The security considerations of [RFC5647] apply.

5. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their

features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

The following example projects maintain an implementation of this protocol:

OpenSSH OpenSSH is the originating implementation of this modes and has supported them since 2013.

Website: https://www.openssh.com/

PuTTY PuTTY is a popular SSH client implementation for multiple platforms that added support for these modes in 2022.

Website: https://www.chiark.greenend.org.uk/~sgtatham/putty/

Paramiko Paramiko is a SSH client and server implementation in the Python programming language. It has supported the these modes modifications since 2024.

Website: https://www.paramiko.org/

Golang x/crypto/ssh The Go programming language project has supported these modes in its external "x" repository since 2023.

Website: https://pkg.go.dev/golang.org/x/crypto/ssh

libssh libssh has implemented these modes since 2018.

Website: https://libssh.org/

Russh Russsh has implemented these modes since 2022.

Website: https://github.com/Eugeny/russh

This list is not exhaustive.

6. References

6.1. 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>.

[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>.

[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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[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>.

[I-D.ietf-sshm-chacha20-poly1305] Miller, D., Tatham, S., and S. Josefsson, "Secure Shell (SSH) authenticated encryption cipher: chacha20-poly1305", Work in Progress, Internet-Draft, draft-ietf-sshm-chacha20-poly1305-01, 17 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-sshm-chacha20-poly1305-01>.

6.2. Informative References

[IANA-SSH-EXT] IANA, "Encryption Algorithm Names", <https://www.iana.org/assignments/ssh-parameters/>.

Acknowledgments

These modes were initially specified and implemented in OpenSSH by Markus Friedl.

Author’s Address

Damien Miller
OpenSSH
Email: djm@openssh.com
URI: https://www.openssh.com/