Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 November 2026

A. Harrison
A. Benhase
Cisco
P. Kampanakis
AWS

27 May 2026

Module-Lattice Key Exchange in SSH
draft-harrison-sshm-mlkem-02

Abstract

This document defines pure post-quantum key exchange methods based on Module-lattice post-quantum key encapsulation schemes for use in the SSH Transport Layer Protocol.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://alharrison.github.io/ssh_mlkem_draft/draft-harrison-mlkem-ssh.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-harrison-sshm-mlkem/.

Source for this draft and an issue tracker can be found at https://github.com/alharrison/ssh_mlkem_draft.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Secure Shell (SSH) [RFC4251] is a secure remote login protocol. The key exchange protocol described in [RFC4253] supports an extensible set of methods. The security of traditional key exchange methods used in Secure Shell (SSH) [RFC4251] relies on the algorithms being too computationally complex to be broken. The development of quantum computers poses a threat to the complexity of these algorithms. Given sufficiently powerful quantum computers, these traditional algorithms would be vulnerable to attack. Additionally, the threat of "harvest-now-decrypt-later" attacks could creates a risk in the current landscape before sufficiently powerful quantum computers are available. In this attack, the data would be collected and decrypted by these quantum computers at a later date.

This document addresses the problem by proposing the use of post-quantum key encapsulation mechanisms (KEMs) to extend the SSH [RFC4253] key exchange. [I-D.draft-ietf-sshm-mlkem-hybrid-kex] introduces ML-KEM in PQ/T Hybrid mode [draft-ietf-pquip-pqt-hybrid-terminology] which combines the shared secrets established by an ECDH and a ML-KEM key exchange. This document uses ML-KEM in a single-algorithm scheme without combining it with a traditional ECDH exchange.

In the context of the [NIST_PQ], key exchange algorithms are formulated as key encapsulation mechanisms (KEMs), which consist of three algorithms:

* 'KeyGen() -> (pk, sk)': A probabilistic key generation algorithm, which generates a public key 'pk' and a secret key 'sk'.

* 'Encaps(pk) -> (ct, ss)': A probabilistic encapsulation algorithm, which takes as input a public key 'pk' and outputs a ciphertext 'ct' and shared secret 'ss'.

* 'Decaps(sk, ct) -> ss': A decapsulation algorithm, which takes as input a secret key 'sk' and ciphertext 'ct' and outputs a shared secret 'ss', or in some cases a distinguished error value.

The main security property for KEMs is indistinguishability under adaptive chosen ciphertext attack (IND-CCA2), which means that shared secret values should be indistinguishable from random strings even given the ability to have arbitrary ciphertexts decapsulated. IND-CCA2 corresponds to security against an active attacker, and the public key / secret key pair can be treated as a long-term key or reused. A weaker security notion is indistinguishability under chosen plaintext attack (IND-CPA), which means that the shared secret values should be indistinguishable from random strings given a copy of the public key. IND-CPA roughly corresponds to security against a passive attacker, and sometimes corresponds to one-time key exchange.

The post-quantum KEM discussed in this document is ML-KEM which is based on CRYSTALS-KYBER. [FIPS203] standardized the ML-KEM scheme in 2024 with three parameter variants, ML-KEM-512, ML-KEM-768, ML-KEM-1024. ML-KEM is a NIST approved mechanism that is believed to be secure against an attacker with a quantum computer.

2. Conventions and Definitions

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.

2.1. ML-KEM Key Exchange Message Numbers

When using ML-KEM as the Key Exchange Method, the following private namespace message numbers are defined in this document: #define SSH_MSG_KEX_KEM_INIT 30 #define SSH_MSG_KEX_KEM_REPLY 31

3. Key Exchange Method: ML-KEM

The client sends SSH_MSG_KEX_KEM_INIT with the following structure:

```
byte      SSH_MSG_KEX_KEM_INIT
string    C_INIT
```

where C_INIT is the ephemeral client ML-KEM public key (C_PK). C_PK represents the public key 'pk' of the client's KeyGen.

The server sends SSH_MSG_KEX_KEM_REPLY with the following structure:

```
byte      SSH_MSG_KEX_KEM_REPLY
string    K_S, server's public host key
string    S_REPLY
string    The signature of hash 'H'
```

where S_REPLY is the ML-KEM ciphertext (S_CT) from the encapsulation of the client's ML-KEM ephemeral public key.

C_PK and S_CT are used to establish the shared secret, K_PQ. K_PQ is the post-quantum shared secret decapsulated from S_CT. Before decapsulating, the client MUST check if the ciphertext S_CT length matches the selected ML-KEM variant. The client MUST abort using a disconnect message (SSH_MSG_DISCONNECT) with a SSH_DISCONNECT_KEY_EXCHANGE_FAILED as the reason if any of the 3 checks specified in Section 7.3 of FIPS 203 fail.

The derivation of encryption keys is done from the shared secret K_PQ according to Section 7.2 in [RFC4253] with a modification on the exchange hash H. The hash H is the result of computing the HASH, where HASH is the hash algorithm specified in the named key exchange method name, over the concatenation of the following

```
string V_C, client identification string (CR and LF excluded)
string V_S, server identification string (CR and LF excluded)
string I_C, payload of the client's SSH_MSG_KEXINIT
string I_S, payload of the server's SSH_MSG_KEXINIT
string K_S, server's public host key
string C_INIT, client message octet string
string S_REPLY, server message octet string
string K_PQ, SSH ML-KEM shared secret
```

3.1. ML-KEM Key Exchange Method Names

The ML-KEM key exchange method names defined in this document (to be used in SSH_MSG_KEXINIT [RFC4253]) are

mlkem512-sha256
mlkem768-sha256
mlkem1024-sha384

Below we define

3.1.1. mlkem512-sha256

mlkem512-sha256 defines the ml-kem-512 C_PK public key and ciphertext S_CT from the client and server respectively which are encoded as octet strings. The K_PQ shared secret is decapsulated from the ciphertext S_CT using the client post-quantum KEM private key as defined in [FIPS203]. K_PQ is encoded as mpint [RFC4251].

The HASH function used in this key exchange [RFC4253] is SHA-256 nist-sha2 [RFC6234]

3.1.2. mlkem768-sha256

mlkem768-sha256 defines the ml-kem-768 C_PK public key and ciphertext S_CT from the client and server respectively which are encoded as octet strings. The K_PQ shared secret is decapsulated from the ciphertext S_CT using the client post-quantum KEM private key as defined in [FIPS203]. K_PQ is encoded as mpint [RFC4251].

The HASH function used in this key exchange [RFC4253] is SHA-256 nist-sha2 [RFC6234]

3.1.3. mlkem1024-sha384

mlkem1024-sha384 defines the ml-kem-1024 C_PK public key and ciphertext S_CT from the client and server respectively which are encoded as octet strings. The K_PQ shared secret is decapsulated from the ciphertext S_CT using the client post-quantum KEM private key as defined in [FIPS203]. K_PQ is encoded as mpint [RFC4251].

The HASH function used in this key exchange [RFC4253] is SHA-384 nist-sha2 [RFC6234]

4. Security Considerations

The security of ML-KEM is based on the presumed difficulty of solving the Module Learning With Errors (MLWE) problem, based on the computational problems in module lattices. [FIPS203]

5. IANA Considerations

IANA is requested to register new method names "mlkem512-sha256", "mlkem768-sha256", "mlkem1024-sha384", and to be registered by IANA in the "Key Exchange Method Names" registry for SSH [IANA-SSH] with a reference field to this RFC and the "OK to implement" field of "MAY".

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/rfc/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/rfc/rfc4251>.

[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/rfc/rfc4253>.

[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/rfc/rfc8174>.

6.2. Informative References

[FIPS203] National Institute of Standards and Technology (NIST), "Module-Lattice-based Key-Encapsulation Mechanism Standard", FIPS PUB 203, August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[I-D.draft-ietf-sshm-mlkem-hybrid-kex] Kampanakis, P., Stabila, D., and T. Hansen, "PQ/T Hybrid Key Exchange in SSH", n.d., <https://datatracker.ietf.org/doc/draft-ietf-sshm-mlkem-hybrid-kex/>.

[IANA-SSH] IANA, "Secure Shell (SSH) Protocol Parameters", 2021, <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml>.

[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/rfc/rfc6234>.

Acknowledgments

Open Quantum Safe has an existing implementation of ML-KEM based key encapsulation methods in all three parameter variants. Their fork of OpenSSH (OQS-SSH) contains an implementation using these algorithms for SSH key exchange algorithms. The authors would like thank Open Quantum Safe for their example implementations of postquantum algorithms.

Authors’ Addresses

Alexander Harrison
Cisco
Email: aleharri@cisco.com

Andrew Benhase
Cisco
Email: abenhase@cisco.com

Panos Kampanakis
AWS
Email: kpanos@amazon.com