Secure Shell (SSHM)
Internet-Draft
Intended status: Standards Track
Expires: 30 October 2026

D. Miller
OpenSSH

28 April 2026

ML-DSA 65/Ed255192 Composite Signatures in SSH
draft-miller-sshm-mldsa65-ed25519-composite-sigs-00

Abstract

This document specifies the integration of the MLDSA65-Ed25519-SHA512 composite signature scheme into the Secure Shell (SSH) protocol. This scheme combines the post-quantum Module-Lattice Digital Signature Algorithm (ML-DSA) with the classical elliptic curve Ed25519 signature algorithm to provide security against both quantum and classical adversaries.

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 30 October 2026.

Table of Contents

1. Introduction

The Secure Shell (SSH) protocol [RFC4251] is a protocol for secure remote login and other secure network services over an untrusted network.

The expected arrival of quantum computing poses a threat to traditional asymmetric cryptography. Post-quantum algorithms such as ML-DSA [FIPS204] have been developed to remain secure against a quantum computer-equipped adversary. However, given the relative novelty of the post-quantum algoritms, it is desirable to use composite schemes that combine post-quantum and classical algorithms to protect against quantum attacks while promising to be "no worse" than the chosen classical algorithm should a vulnerability be discovered in the post-quantum algorithm or its implementation.

This document specifies a composite signature scheme for SSH, mapping the COMPSIG-MLDSA65-Ed25519-SHA512 scheme defined in [I-D.ietf-lamps-pq-composite-sigs] to the SSH protocol [RFC4253] using the identifier ssh-mldsa65-ed25519@openssh.com.

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. Terminology

All encoding data types ("byte", "string", etc.) are as specified in Section 5 of [RFC4251].

All length units are given in bytes unless otherwise specified.

4. The ssh-mldsa65-ed25519@openssh.com Algorithm

The algorithm name for this composite signature scheme is "ssh-mldsa65-ed25519@openssh.com". The underlying key generation, signing and verification primitives are as specified in [I-D.ietf-lamps-pq-composite-sigs], but are also described briefly below.

4.1. Key generation

Keys are generated using the process specified in Section 3.1 of [I-D.ietf-lamps-pq-composite-sigs]. Specifically, ML-DSA and Ed25519 keys are generated separately from seeds and the resultant public for each are combined to form the composite public key. Private keys are combined as seeds and not as expanded keys.

ML-DSA key generation uses the ML-DSA.KeyGen_internal algorithm as defined in [FIPS204] section 6.1 and not the usual ML-DSA.KeyGen algorithm as the latter emits an expanded private key and not the required seed.

Ed25519 key generation is described in Section 5.1.5 of [RFC8032].

Seeds to both ML-DSA and Ed25519 MUST be sourced from a cryptographically sound source, such as a CSPRNG/RBG.

4.1.1. Composite raw public key

The resultant public keys from the respective ML-DSA and Ed25519 key generation algorithms are combined by simple concatenation to form a composite public key. Specifically it consists of the raw ML-DSA-65 public key (1952 bytes) and the Ed25519 public key (32 bytes), for a total of 1984 bytes:

```
byte[1952] mldsa_pk
byte[32]   ed25519_pk
```

This 1984-byte combination is referred to as "composite_public_key" henceforth in this document.

4.1.2. Composite raw private key

The composite private key is also a concatenation, but its elements are the private key seeds for the respective algorithms. Each of these seeds is 32 bytes.

```
byte[32]  mldsa_seed
byte[32]  ed25519_seed
```

This 64-byte combination is referred to as "composite_private_key" henceforth in this document.

4.2. SSH Public Key Format

The "ssh-mldsa65-ed25519@openssh.com" public key as it appears in the SSH protocol has the following format:

```
string    "ssh-mldsa65-ed25519@openssh.com"
string    composite_public_key
```

Where "composite_public_key" is as described in Section 4.1.1.

4.3. SSH Private Key Format

When adding a key to an agent [I-D.ietf-sshm-ssh-agent] using the SSH_AGENTC_ADD_IDENTITY or SSH_AGENTC_ADD_ID_CONSTRAINED messages, the key data MUST have the following format:

```
string    "ssh-mldsa65-ed25519@openssh.com"
string    composite_public_key
string    composite_private_key
```

Where "composite_public_key" is as described in Section 4.1.1 and "composite_private_key" is as described in Section 4.1.2.

This private key serialisation may also be used as the basis of on-disk key formats, though these are beyond the scope of this document.

4.4. Signatures

4.4.1. Signature Generation

To sign a message M, the signer first constructs a message representative M' as follows:

```
M' = Prefix || Label || len(ctx) || ctx || SHA512(M)
```

Where:

* Prefix: The ASCII string "CompositeAlgorithmSignatures2025" [43 6F 6D 70 6F 73 69 74 65 41 6C 67 6F 72 69 74 68 6D 53 69 67 6E 61 74 75 72 65 73 32 30 32 35].

* Label: The ASCII string "COMPSIG-MLDSA65-Ed25519-SHA512" [43 4F 4D 50 53 49 47 2D 4D 4C 44 53 41 36 35 2D 45 64 32 35 35 31 39 2D 53 48 41 35 31 32].

* len(ctx): A single byte representing the length of the context string.

* ctx: The context string.

* SHA512(M): The SHA-512 hash of the original message M.

The signer then computes:

* mldsa_sig = ML-DSA-65.Sign(mldsa_sk, M', ctx=Label)

* ed25519_sig = Ed25519.Sign(ed25519_sk, M')

The final composite signature is the concatenation of mldsa_sig and ed25519_sig.

```
byte[3309] mldsa_sig
byte[64]   ed25519_sig
```

This 3373-byte combination is henceforth referred to as "composite_signature".

4.4.2. Signature Verification

To verify a signature, the verifiere reconstructs M' using the same process as defined for generation. The signature is valid if and only if both underlying signature verifications succeed:

* ML-DSA-65.Verify(mldsa_pk, M', mldsa_sig, ctx=Label)

* Ed25519.Verify(ed25519_pk, M', ed25519_sig)

4.4.3. SSH signature wire encoding

Signatures are represented in SSH using the following format:

```
    string    "ssh-mldsa65-ed25519@openssh.com"
    string    composite_signature
```

Where "composite_signature" is as defined in Section 4.4.1.

4.4.4. Signture contexts

The COMPSIG-MLDSA65-Ed25519-SHA512 signature algorithm, like its underlying ML-DSA component, accepts a Context parameter that may be used to enforce domain separation between signatues.

TODO TODO TODO In all cases in the SSH protocol, this context value is the empty string, but if we ever change our minds then this is where we'd describe what contexts are used for each signature/verification operation. TODO TODO TODO.

5. Security Considerations

The security of this composite scheme depends on the strength of both component algorithms. An adversary must break both ML-DSA-65 and Ed25519 to forge a signature.

Implementations MUST NOT reuse component key material between composite and non-composite keys, or between multiple composite keys.

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>.

[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>.

[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>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[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-lamps-pq-composite-sigs] Ounsworth, M., Gray, J., Pala, M., Klauテ殤er, J., and S. Fluhrer, "Composite Module-Lattice-Based Digital Signature Algorithm (ML-DSA) for use in X.509 Public Key Infrastructure", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-sigs-19, 21 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-19>.

[FIPS204] "Module-lattice-based digital signature standard", National Institute of Standards and Technology (U.S.), FIPS 204, DOI 10.6028/nist.fips.204, August 2024, <https://doi.org/10.6028/nist.fips.204>.

6.2. Informative References

[I-D.ietf-sshm-ssh-agent] Miller, D., "SSH Agent Protocol", Work in Progress, Internet-Draft, draft-ietf-sshm-ssh-agent-15, 28 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-sshm-ssh-agent-15>.

Author’s Address

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