sshm
Internet-Draft
Intended status: Standards Track
Expires: 9 July 2026

S. Sun
L. Prabel
Huawei

5 January 2026

Composite ML-DSA Signatures for SSH
draft-sun-ssh-composite-sigs-02

Abstract

This document describes the use of PQ/T composite signatures for the Secure Shell (SSH) protocol. The composite signatures described combine ML-DSA as the post-quantum part and the elliptic curve signature schemes ECDSA, Ed25519 and Ed448 as the traditional part.

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://example.com/LATEST. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-sun-ssh-composite-sigs/.

Discussion of this document takes place on the Secure Shell Maintenance (sshm) Working Group mailing list (mailto:ssh@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ssh/. Subscribe at https://www.ietf.org/mailman/listinfo/ssh/.

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

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 9 July 2026.

Table of Contents

1. Introduction

The development of quantum computers has raised concern towards traditional asymmetric cryptographic algorithms. A Cryptographically Relevant Quantum Computer (CRQC) will break RSA and elliptic curve signature schemes. There is a need to migrate to quantum-resistant signature schemes.

Recently, NIST publised the ML-DSA [FIPS204] algorithm, which is a post-quantum signature scheme. However, when using relatively new cryptographic schemes, the lack of maturing time makes people worry. Many hybrid solutions are thus proposed, which combine a traditional algorithm with a post-quantum algorithm.

[FIPS204] defines both pure ML-DSA and pre-hash ML-DSA. This document only uses pure ML-DSA.

This document describes how to combine ML-DSA with the elliptic curve signature schemes ECDSA, Ed25519 and Ed448 for authentication in the SSH [RFC4251] protocol.

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.

This document is consistent with the terminology for hybrid signatures defined in [I-D.draft-ietf-pquip-pqt-hybrid-terminology].

The key and signature formats follows the notation introduced in [RFC4251], Section 3, and the string data type format follows the notation from [RFC4251], Section 5.

3. Composite Algorithms

A composite algorithm has one post-quantum algorithm, and one traditional algorithm.

3.1. Composite Key Generation

Composite public and private keys are generated by calling the key generation functions of the two component algorithms and concatenating the keys in an order given by the registered composite algorithm.

For the composite algorithms described in this document, the Key Generation process is as follows:

1. Generate component keys

```
(mldsaPK, mldsaSK) = ML-DSA.KeyGen()
(tradPK, tradSK) = ECCSigAlg.KeyGen()
```

2. Check for component key generation failure

```
if NOT (mldsaPK, mldsaSK) or NOT (tradPK, tradSK):
    output "Key generation error"
```

3. Serialize keys into composite form

```
Composite Public Key  <- SerializePublicKey(mldsaPK, tradPK)
Composite Private Key <- SerializePrivateKey(mldsaSK, tradSK)
```

It makes use of the serialization routines from [I-D.draft-ietf-lamps-pq-composite-sigs] to obtain the byte string encodings of the composite public and private keys.

ECCSigAlg is an elliptic curve signature scheme, i.e., ECDSA, Ed25519 or Ed448.

3.2. Composite Sign

A composite signature’s value MUST include two signature components and MUST be in the same order as the components from the corresponding signing key.

For the composite algorithms described in this document, the signature process of a message M follows Section 4.2 of [I-D.draft-ietf-lamps-pq-composite-sigs], with an empty application context string:

1. Compute the Message representative M’

```
M’ <- Prefix || Label || 0x00 || PH(M)
```

2. Get the component keys

```
(mldsaSK, tradSK) = DeserializePrivateKey(sk)
```

3. Generate the two component signatures

```
sig_1 <- ML-DSA.Sign(mldsaSK, M’, ctx=Label)
sig_2 <- ECCSigAlg.Sign(tradSK, M’)
```

4. If either ML-DSA.Sign() or ECCSigAlg.Sign() return an error, then this process MUST return an error.

```
if NOT sig_1 or NOT sig_2:
  output "Signature generation error"
```

5. Output the encoded composite signature value.

```
CompositeSignature <- SerializeSignatureValue(sig_1, sig_2)
return CompositeSignature
```

It makes use of the serialization routines from [I-D.draft-ietf-lamps-pq-composite-sigs] to obtain the byte string encodings of the composite signature.

The prefix "Prefix" string is defined as in [I-D.draft-ietf-lamps-pq-composite-sigs] as the byte encoding of the string "CompositeAlgorithmSignatures2025", which in hex is 436F6D706F73697465416C676F726974686D5369676E617475726573323032 35. It can be used by a traditional verifier to detect if the composite signature has been stripped apart.

The label "Label" is defined in the same way as [I-D.draft-ietf-lamps-pq-composite-sigs] and is passed as a context argument into the underlying ML-DSA component algorithm. The label values, specific to each composite algorithm, can be found in Table 1.

Key Format identifier	Label (in ASCII)	Label (in Hex encoding)
ssh-mldsa44-es256	COMPSIG-MLDSA44-ECDSA-P256-SHA256	434F4D505349472D4D4C44534134342D45434453412D503235362D534841323536
ssh-mldsa65-es256	COMPSIG-MLDSA65-ECDSA-P256-SHA512	434F4D505349472D4D4C44534136352D45434453412D503235362D534841353132
ssh-mldsa87-es384	COMPSIG-MLDSA87-ECDSA-P384-SHA512	434F4D505349472D4D4C44534138372D45434453412D503338342D534841353132
ssh-mldsa44-ed25519	COMPSIG-MLDSA44-Ed25519-SHA512	434F4D505349472D4D4C44534134342D45643235353139 2D534841353132
ssh-mldsa65-ed25519	COMPSIG-MLDSA65-Ed25519-SHA512	434F4D505349472D4D4C44534136352D45643235353139 2D534841353132
ssh-mldsa87-ed448	COMPSIG-MLDSA87-Ed448-SHAKE256	434F4D505349472D4D4C44534138372D45643434382D53 48414B45323536

Table 1: Composite Label Values

3.3. Composite Verify

The Verify algorithm MUST validate a signature only if all component signatures are successfully validated.

1. Get the component keys and signatures

```
(mldsaPK, tradPK) <- DeserializePublicKey(pk)
(sig_1, sig_2) <- DeserializeSignatureValue(sig)
```

2. Compute the message representative M’

```
M’ <- Prefix || Label || 0x00 || PH(M)
```

3. Check each component signature individually, according to its algorithm specification.

```
    if NOT ML-DSA.Verify(mldsaPK, M', ctx=Label)
        output "Invalid signature"
    if NOT ECCSigAlg.Verify(tradPK, M')
        output "Invalid signature"
    if all succeeded, then
        output "Valid signature"
```

It makes use of the serialization routines from [I-D.draft-ietf-lamps-pq-composite-sigs] to obtain the component public keys and the component signatures.

4. Public Key Algorithm

This section gives the concrete composite signature algorithms and their component algorithms. Their usage within SSH follows Section 6.6 of [RFC4253].

The following table defines a list of algorithms associated with specific PQ/T combinations.

Key Format Identifier	First Algorithm	Second Algorithm	Pre-Hash	Description
ssh-mldsa44-es256	ML-DSA-44	ecdsa-with-SHA256 with secp256r1	SHA256	Composite Signature with ML-DSA-44 and ECDSA using P-256 curve and SHA-256
ssh-mldsa65-es256	ML-DSA-65	ecdsa-with-SHA256 with secp256r1	SHA512	Composite Signature with ML-DSA-65 and ECDSA using P-256 curve and SHA-256
ssh-mldsa87-es384	ML-DSA-87	ecdsa-with-SHA384 with secp384r1	SHA512	Composite Signature with ML-DSA-87 and ECDSA using P-384 curve and SHA-384
ssh-mldsa44-ed25519	ML-DSA-44	Ed25519	SHA512	Composite Signature with ML-DSA-44 and Ed25519
ssh-	ML-DSA-65	Ed25519	SHA512	Composite

mldsa65-ed25519				Signature with ML-DSA-65 and Ed25519
ssh-mldsa87-ed448	ML-DSA-87	Ed448	SHAKE256	Composite Signature with ML-DSA-87 and Ed448

Table 2: Composite ML-DSA Signature Algorithms for SSH

5. Public Key Format

The key format for all parameter sets defined in this document follows the encoding pattern from [RFC4253], Section 6.6.

string identifier

string key

The 'identifier' is the key format identifier given in Table 2.

The 'key' is the composite public key generated as defined in Section 3.1. It is the concatenation of the public keys of the component schemes.

For ML-DSA, the public keys are defined in [FIPS204].

For ECDSA with curves secp256r1 or secp384r1, the public keys are defined in [RFC5656], Section 3.1. The public key is encoded from an elliptic curve point into an octet string as defined in Section 2.3.3 of [SEC1]; point compression MAY be used.

For Ed25519 and Ed448, the public keys are defined in [RFC8709], Section 6.

The "ssh-mldsa44-es256" key format has the following encoding:

string ssh-mldsa44-es256

string key

Here, 'key' is the concatenation of the 1312-octet ML-DSA-44 public key and the ECDSA public key using the secp256r1 curve.

The "ssh-mldsa65-es256" key format has the following encoding:

string ssh-mldsa65-es256

string key

Here, 'key' is the concatenation of the 1952-octet ML-DSA-65 public key and the ECDSA public key using the secp256r1 curve.

The "ssh-mldsa87-es384" key format has the following encoding:

string ssh-mldsa87-es384

string key

Here, 'key' is the concatenation of the 2592-octet ML-DSA-87 public key and the ECDSA public key using the secp384r1 curve.

The "ssh-mldsa44-ed25519" key format has the following encoding:

string ssh-mldsa44-ed25519

string key

Here, 'key' is the concatenation of the 1312-octet ML-DSA-44 public key and the 32-octet Ed25519 public key.

The "ssh-mldsa65-ed25519" key format has the following encoding:

string ssh-mldsa65-ed25519

string key

Here, 'key' is the concatenation of the 1952-octet ML-DSA-65 public key and the 32-octet Ed25519 public key.

The "ssh-mldsa87-ed448" key format has the following encoding:

string ssh-mldsa87-ed448

string key

Here, 'key' is the concatenation of the 2592-octet ML-DSA-87 public key and the 57-octet Ed448 public key.

6. Signature Format

The signature format for all parameter sets defined in this document follows the encoding pattern defined in Section 6.6 of [RFC4253].

string identifier

string signature

The ’identifier’ is the key format identifier given in Section 4.

The ’signature’ is the composite signature generated as defined in Section 3.1. It is the concatenation of the signatures of the component schemes.

For ML-DSA, the signatures are defined in [FIPS204].

For ECDSA with curves secp256r1 and secp384r1, the signatures and their encodings are defined in [RFC5656], Section 3.1.2.

For Ed25519 and Ed448, the signature are defined in [RFC8709], Section 6.

The "ssh-mldsa44-es256" signature format has the following encoding:

string ssh-mldsa44-es256

string signature

Here, ’signature’ is the concatenation of the 2420-octet ML-DSA-44 signature and the ECDSA signature using the secp256r1 curve.

The "ssh-mldsa65-es256" signature format has the following encoding:

string ssh-mldsa65-es256

string signature

Here, ’signature’ is the concatenation of the 3309-octet ML-DSA-65 signature and the ECDSA signature using the secp256r1 curve.

The "ssh-mldsa87-es384" signature format has the following encoding:

string ssh-mldsa87-es384

string signature

Here, ’signature’ is the concatenation of the 4627-octet ML-DSA-44 signature and the ECDSA signature using the secp384r1 curve.

The "ssh-mldsa44-ed25519" signature format has the following encoding:

string ssh-mldsa44-ed25519

string signature

Here, ’signature’ is the concatenation of the 2420-octet ML-DSA-44 signature and the 64-octet Ed25519 signature.

The "ssh-mldsa65-ed25519" signature format has the following encoding:

string ssh-mldsa65-ed25519

string signature

Here, ’signature’ is the concatenation of the 3309-octet ML-DSA-44 signature and the 64-octet Ed25519 signature.

The "ssh-mldsa87-ed448" signature format has the following encoding:

string ssh-mldsa87-ed448

string signature

Here, ’signature’ is the concatenation of the 4627-octet ML-DSA-44 signature and the 114-octet Ed448 signature.

7. Security Considerations

The Security Considerations section of [RFC4251] also applies to this document.

The user can refer to [FIPS204] for security issues related to the ML-DSA post-quantum component of the composite algorithm and to the Security Considerations sections of [RFC5656] and [RFC8709] for the traditional component.

For the specific security issues raising from the use of a hybrid composite signature scheme, the user can refer to [I-D.draft-ietf-pquip-hybrid-signature-spectrums].

For more information about hybrid composite signature schemes and the different hybrid combinations that appear in this document, the user can read [I-D.draft-ietf-lamps-pq-composite-sigs].

8. IANA Considerations

IANA is requested to add the following entries to "Public Key Algorithm Names" in the "Secure Shell (SSH) Protocol Parameters" registry [IANA.SSH]:

Public Key Algorithm Name	Key Format	Reference
ssh-mldsa44-es256	ssh-mldsa44-es256	THIS-RFC
ssh-mldsa65-es256	ssh-mldsa65-es256	THIS-RFC
ssh-mldsa87-es384	ssh-mldsa87-es384	THIS-RFC
ssh-mldsa44-ed25519	ssh-mldsa44-ed25519	THIS-RFC
ssh-mldsa65-ed25519	ssh-mldsa65-ed25519	THIS-RFC
ssh-mldsa87-ed448	ssh-mldsa87-ed448	THIS-RFC

Table 3: SSH Public Key Code Points

9. References

9.1. 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>.

[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/rfc/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/rfc/rfc8174>.

[RFC8709] Harris, B. and L. Velvindron, "Ed25519 and Ed448 Public Key Algorithms for the Secure Shell (SSH) Protocol", RFC 8709, DOI 10.17487/RFC8709, February 2020, <https://www.rfc-editor.org/rfc/rfc8709>.

9.2. Informative References

[FIPS204] National Institute of Standards and Technology (NIST), "Module-Lattice-Based Digital Signature Standard", August 2024, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS204.pdf>.

[I-D.draft-ietf-lamps-pq-composite-sigs] Ounsworth, M., Gray, J., Pala, M., Klauテ殤er, J., and S. Fluhrer, "Composite ML-DSA for use in X.509 Public Key Infrastructure", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-sigs-13, 31 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-sigs-13>.

[I-D.draft-ietf-pquip-hybrid-signature-spectrums] Bindel, N., Hale, B., Connolly, D., and F. D, "Hybrid signature spectrums", Work in Progress, Internet-Draft, draft-ietf-pquip-hybrid-signature-spectrums-07, 20 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-hybrid-signature-spectrums-07>.

[I-D.draft-ietf-pquip-pqt-hybrid-terminology] D, F., P, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-ietf-pquip-pqt-hybrid-terminology-06, 10 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqt-hybrid-terminology-06>.

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

[SEC1] Standards for Efficient Cryptography Group, "Elliptic Curve Cryptography", May 2009, <http://www.secg.org/download/aid-780/sec1-v2.pdf>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Sun Shuzhou
Huawei
Email: sunshuzhou@huawei.com

Lucas Prabel
Huawei
Email: lucas.prabel@huawei.com