TLS
Internet-Draft
Intended status: Standards Track
Expires: 15 November 2026

T. Reddy
Nokia
T. Hollebeek
DigiCert
J. Gray
Entrust
S. Fluhrer
Cisco Systems
D. V. Geest
CryptoNext Security

14 May 2026

Use of Composite ML-DSA in TLS 1.3
draft-reddy-tls-composite-mldsa-10

Abstract

Compositing the post-quantum ML-DSA signature with traditional signature algorithms provides protection against potential breaks or critical bugs in ML-DSA or the ML-DSA implementation. This document specifies how such a composite signature can be formed using ML-DSA with RSA-PKCS#1 v1.5, RSA-PSS, ECDSA, Ed25519, and Ed448 to provide authentication in TLS 1.3, including use in certificates.

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 15 November 2026.

Copyright Notice

Table of Contents

1. Introduction

The advent of quantum computing poses a significant threat to current cryptographic systems. Traditional cryptographic algorithms such as RSA, Diffie-Hellman, DSA, and their elliptic curve variants are vulnerable to quantum attacks. During the transition to post-quantum cryptography (PQC), there is considerable uncertainty regarding the robustness of both existing and new cryptographic algorithms. While we can no longer fully trust traditional cryptography, we also cannot immediately place complete trust in post-quantum replacements until they have undergone extensive scrutiny and real-world testing to uncover and rectify potential implementation flaws.

Unlike previous migrations between cryptographic algorithms, the decision of when to migrate and which algorithms to adopt is far from straightforward. Even after the migration period, it may be advantageous for an entity's cryptographic identity to incorporate multiple public-key algorithms to enhance security.

Cautious implementers may opt to combine cryptographic algorithms in such a way that an attacker would need to break all of them simultaneously to compromise the protected data. These mechanisms are referred to as Post-Quantum/Traditional (PQ/T) Hybrids [RFC9794].

One practical way to implement a hybrid signature scheme is through a composite signature algorithm. In this approach, the composite signature consists of two signature components, each produced by a different signature algorithm. A composite key is treated as a single key that performs a single cryptographic operation such as key generation, signing and verification by using its internal sequence of component keys as if they form a single key.

Certain jurisdictions are already recommending or mandating that PQC lattice schemes be used exclusively within a PQ/T hybrid framework. The use of composite schemes provides a straightforward implementation of hybrid solutions compatible with (and advocated by) some governments and cybersecurity agencies [BSI2021].

ML-DSA [FIPS204] is a post-quantum signature scheme standardised by NIST. It is a module-lattice based scheme.

This memo specifies how a composite ML-DSA can be negotiated for authentication in TLS 1.3 via the "signature_algorithms" and "signature_algorithms_cert" extensions. Hybrid signatures provide additional safety by ensuring protection even if vulnerabilities are discovered in one of the constituent algorithms. For deployments that cannot easily tweak configuration or effectively enable/disable algorithms, a composite signature combining PQC signature algorithm with a traditional signature algorithm offers the most viable solution.

The rationale for this approach is based on the limitations of fallback strategies. For example, if a traditional signature system is compromised, reverting to a PQC signature algorithm would prevent attackers from forging new signatures that are no longer accepted. However, such a fallback process leaves systems exposed until the transition to the PQC signature algorithm is complete, which can be slow in many environments. In contrast, using hybrid signatures from the start mitigates this issue, offering robust protection and encouraging faster adoption of PQC.

Further, zero-day vulnerabilities, where an exploit is discovered and used before the vulnerability is publicly disclosed, highlights this risk. The time required to disclose such attacks and for organizations to reactively switch to alternative algorithms can leave systems critically exposed. By the time a secure fallback is implemented, attackers may have already caused irreparable damage. Adopting hybrid signatures preemptively helps mitigate this window of vulnerability, ensuring resilience even in the face of unforeseen threats.

1.1. Conventions and 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. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

This document is consistent with the terminology defined in [RFC9794]. It defines composites as:

> _Composite Cryptographic Element_: A cryptographic element that incorporates multiple component cryptographic elements of the same type in a multi-algorithm scheme.

In this document, “composite ML-DSA” refers to a composite ML-DSA signature scheme as defined in [I-D.ietf-lamps-pq-composite-sigs].

2. ML-DSA SignatureSchemes

As defined in [RFC8446], the SignatureScheme namespace is used for the negotiation of signature schemes for authentication via the "signature_algorithms" and "signature_algorithms_cert" extensions. This document adds new SignatureScheme values for composite ML-DSA as follows.

```
  enum {
    /* ECDSA-based Composite */
    mldsa44_ecdsa_secp256r1_sha256 (TBD1),
    mldsa65_ecdsa_secp256r1_sha512 (TBD2),
    mldsa65_ecdsa_secp384r1_sha512 (TBD3),
    mldsa87_ecdsa_secp384r1_sha512 (TBD4),

    /* EdDSA-based Composite */
    mldsa44_ed25519_sha512 (TBD5),
    mldsa65_ed25519_sha512 (TBD6),
    mldsa87_ed448_shake256 (TBD7),

    /* RSA-PKCS1-based Composite (for signature_algorithms_cert ONLY) */
    mldsa44_rsa2048_pkcs15_sha256 (TBD8),
    mldsa65_rsa3072_pkcs15_sha512 (TBD9),
    mldsa65_rsa4096_pkcs15_sha512 (TBD10),

    /* RSA-PSS-based Composite */
    mldsa44_rsa2048_pss_sha256 (TBD11),
    mldsa65_rsa3072_pss_sha512 (TBD12),
    mldsa87_rsa3072_pss_sha512 (TBD13),
    mldsa65_rsa4096_pss_sha512 (TBD14),
    mldsa87_rsa4096_pss_sha512 (TBD15)

  } SignatureScheme;
```

Composite ML-DSA is treated as an opaque signature algorithm by TLS, similar to Ed25519 and Ed448, which use the pure (non-prehashed) forms specified in TLS 1.3 as "PureEdDSA" algorithms (Section 4.2.3 of [RFC8446]). The SignatureScheme names defined in this document (for example, mldsa44_ecdsa_secp256r1_sha256) mirror the algorithm names in [I-D.ietf-lamps-pq-composite-sigs] (for example, id-MLDSA44-ECDSA-P256-SHA256). TLS implementors do not need to be aware of these internal details; for a full description of the composite algorithm construction, see Sections 3.2 and 3.3 of [I-D.ietf-lamps-pq-composite-sigs].

SignatureScheme names are used only as identifiers for negotiation and registry purposes and do not imply TLS-level processing semantics.

In composite ML-DSA schemes, the SignatureScheme name encodes the PQC component (for example, mldsa44), the traditional signature algorithm and curve (for example, ecdsa_secp256r1), and the internal prehash function (for example, sha256) used by the Composite ML-DSA algorithm prior to generating the individual component signatures, as defined in [I-D.ietf-lamps-pq-composite-sigs]. This identification is for algorithm selection and interoperability purposes only and does not imply any TLS-level processing of the traditional component.

The explicit RSA key size (for example, RSA2048, RSA3072, or RSA4096) is included in the SignatureScheme name solely to uniquely identify the composite algorithm and to align with the composite algorithm definitions in [I-D.ietf-lamps-pq-composite-sigs].

Each entry specifies a unique combination of an ML-DSA parameter set (ML-DSA-44, ML-DSA-65, or ML-DSA-87, as defined in [FIPS204]) and a traditional signature algorithm. The mldsa* identifiers refer to the pure ML-DSA variants and MUST NOT be confused with prehashed variants (for example, HashML-DSA-44).

Sections 3.2 and 3.3 of [I-D.ietf-lamps-pq-composite-sigs] define a context string parameter for signing and verification using Composite ML-DSA. When Composite ML-DSA signature algorithms are used in TLS, both signing and verification MUST use an empty context string. TLS already provides protocol-level domain separation by signing a protocol-specific context string together with the handshake transcript (Section 4.4.3 of [RFC8446]).

When a composite ML-DSA signature scheme defined in this document is negotiated, the TLS 1.3 CertificateVerify signing input constructed as specified in Section 4.4.3 of [RFC8446] is signed using the negotiated composite ML-DSA SignatureScheme, as specified in [I-D.ietf-lamps-pq-composite-sigs].

Upon receipt of the CertificateVerify message, the peer MUST verify the signature over the locally constructed signing input using the negotiated composite ML-DSA SignatureScheme, in accordance with [I-D.ietf-lamps-pq-composite-sigs].

When a composite ML-DSA SignatureScheme is negotiated, the end-entity certificate presented in the TLS handshake MUST contain a public key compatible with that SignatureScheme, as specified in [I-D.ietf-lamps-pq-composite-sigs].

The schemes defined in this document MUST NOT be used in TLS 1.2 [RFC5246]. A peer that receives ServerKeyExchange or CertificateVerify message in a TLS 1.2 connection with schemes defined in this document MUST abort the connection with an illegal_parameter alert.

3. Signature Algorithm Restrictions

TLS 1.3 removed support for RSASSA-PKCS1-v1_5 [RFC8017] in CertificateVerify messages, opting for RSASSA-PSS instead. Similarly, this document restricts the use of the composite signature algorithms mldsa44_rsa2048_pkcs15_sha256, mldsa65_rsa3072_pkcs15_sha512, and mldsa65_rsa4096_pkcs15_sha512 algorithms, defined in [I-D.ietf-lamps-pq-composite-sigs], to the "signature_algorithms_cert" extension. These composite signature algorithms MUST NOT be used with the "signature_algorithms" extension. These values refer solely to signatures which appear in certificates (see Section 4.4.2.2 of [RFC8446]) and are not defined for use in signed TLS handshake messages.

A peer that receives a CertificateVerify message indicating the use of the RSASSA-PKCS1-v1_5 algorithm as one of the component signature algorithms MUST terminate the connection with a fatal illegal_parameter alert.

4. Selection Criteria for Composite Signature Algorithms

The composite signatures specified in the document are a restricted set of cryptographic pairs, chosen from the intersection of two sources:

* The composite algorithm combinations as recommended in [I-D.ietf-lamps-pq-composite-sigs], which specify both PQC and traditional signature algorithms.

* The recommended traditional signature algorithms listed in TLS 1.3.

By limiting algorithm combinations to those defined in both [I-D.ietf-lamps-pq-composite-sigs] and TLS 1.3, this specification ensures that each pair meets established security standards for composite signatures in a post-quantum context, as described in [I-D.ietf-lamps-pq-composite-sigs].

This conservative approach reduces the risk of selecting unsafe or incompatible configurations, promoting security by requiring only trusted and well-vetted pairs. Future updates to this specification may introduce additional algorithm pairs as standards evolve, subject to similar vetting and inclusion criteria.

4.1. Mapping TLS SignatureSchemes to Composite ML-DSA

The following table provides a mapping between the TLS SignatureScheme identifiers defined in this document and the corresponding composite algorithm identifiers defined in [I-D.ietf-lamps-pq-composite-sigs].

TLS SignatureScheme	Composite ML-DSA Algorithm Name
mldsa44_ecdsa_secp256r1_sha256	id-MLDSA44-ECDSA-P256-SHA256
mldsa65_ecdsa_secp256r1_sha512	id-MLDSA65-ECDSA-P256-SHA512
mldsa65_ecdsa_secp384r1_sha512	id-MLDSA65-ECDSA-P384-SHA512
mldsa87_ecdsa_secp384r1_sha512	id-MLDSA87-ECDSA-P384-SHA512
mldsa44_ed25519_sha512	id-MLDSA44-Ed25519-SHA512
mldsa65_ed25519_sha512	id-MLDSA65-Ed25519-SHA512
mldsa87_ed448_shake256	id-MLDSA87-Ed448-SHAKE256
mldsa44_rsa2048_pss_sha256	id-MLDSA44-RSA2048-PSS-SHA256
mldsa65_rsa3072_pss_sha512	id-MLDSA65-RSA3072-PSS-SHA512
mldsa87_rsa3072_pss_sha512	id-MLDSA87-RSA3072-PSS-SHA512
mldsa65_rsa4096_pss_sha512	id-MLDSA65-RSA4096-PSS-SHA512
mldsa87_rsa4096_pss_sha512	id-MLDSA87-RSA4096-PSS-SHA512
mldsa44_rsa2048_pkcs15_sha256	id-MLDSA44-RSA2048-PKCS15-SHA256
mldsa65_rsa3072_pkcs15_sha512	id-MLDSA65-RSA3072-PKCS15-SHA512
mldsa65_rsa4096_pkcs15_sha512	id-MLDSA65-RSA4096-PKCS15-SHA512

Table 1

5. Security Considerations

The security considerations discussed in Section 11 of [I-D.ietf-lamps-pq-composite-sigs] need to be taken into account.

Traditional signature algorithms such as ECDSA, Ed25519, and Ed448 provide existential unforgeability under chosen-message attack (EUF-CMA), which is sufficient for TLS authentication. When used as the traditional component in a composite construction with ML-DSA, these algorithms contribute to defense-in-depth during the transition to post-quantum cryptography, maintaining TLS authentication security as long as at least one component algorithm remains secure.

However, composite signature schemes do not in general preserve strong unforgeability (SUF-CMA) once the traditional component algorithm is broken, for example due to the availability of CRQCs. In such cases, a forged traditional signature component can be combined with a valid post-quantum component to produce a composite signature that verifies successfully, violating SUF. This loss of SUF is inherent to the composite construction and does not impact TLS, which requires only EUF-CMA security from its signature schemes.

TLS clients that support both post-quantum and traditional-only signature algorithms are vulnerable to downgrade attacks. In such scenarios, an attacker with access to a CRQC could forge a traditional server certificate and impersonate the server. If the client continues to accept traditional-only certificates for backward compatibility, it remains exposed to this risk.

While broader deployment of composite or post-quantum certificates will reduce this exposure, clients remain vulnerable unless stricter authentication continuity policies are enforced. A coordinated “flag day” in which all traditional-only certificates are simultaneously phased out is unlikely due to real-world deployment constraints. The continuity mechanism defined in [I-D.sheffer-tls-pqc-continuity] addresses this deployment challenge by allowing clients to cache and enforce a server’s support for post-quantum or composite authentication, thereby preventing fallback to traditional-only authentication in subsequent connections.

6. IANA Considerations

This document requests new entries to the TLS SignatureScheme registry, according to the procedures in Section 6 of [TLSIANA].

Value	Description	Recommended	Reference
TBD1	mldsa44_ecdsa_secp256r1_sha256	N	This document.
TBD2	mldsa65_ecdsa_secp256r1_sha512	N	This document.
TBD3	mldsa65_ecdsa_secp384r1_sha512	N	This document.
TBD4	mldsa87_ecdsa_secp384r1_sha512	N	This document.
TBD5	mldsa44_ed25519_sha512	N	This document.
TBD6	mldsa65_ed25519_sha512	N	This document.
TBD7	mldsa87_ed448_shake256	N	This document.
TBD8	mldsa44_rsa2048_pkcs15_sha256	N	This document.
TBD9	mldsa65_rsa3072_pkcs15_sha512	N	This document.
TBD10	mldsa65_rsa4096_pkcs15_sha512	N	This document.
TBD11	mldsa44_rsa2048_pss_sha256	N	This document.
TBD12	mldsa65_rsa3072_pss_sha512	N	This document.
TBD13	mldsa87_rsa3072_pss_sha512	N	This document.
TBD14	mldsa65_rsa4096_pss_sha512	N	This document.
TBD15	mldsa87_rsa4096_pss_sha512	N	This document.

Table 2

6.1. Restricting Composite Signature Algorithms to the signature_algorithms_cert Extension

IANA is requested to add a footnote indicating that the mldsa44_rsa2048_pkcs15_sha256, mldsa65_rsa3072_pkcs15_sha512, and mldsa65_rsa4096_pkcs15_sha512 algorithms are defined exclusively for use with the signature_algorithms_cert extension and are not intended for use with the signature_algorithms extension.

7. References

7.1. Normative References

[I-D.ietf-lamps-pq-composite-sigs] Ounsworth, M., Gray, J., Pala, M., Klauner, 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>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[TLSIANA] Salowey, J. A. and S. Turner, "IANA Registry Updates for TLS and DTLS", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8447bis-15, 21 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-rfc8447bis-15>.

7.2. Informative References

[BSI2021] Federal Office for Information Security (BSI), "Quantum-safe cryptography - fundamentals, current developments and recommendations", October 2021, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.pdf>.

[FIPS204] "FIPS-204: Module-Lattice-Based Digital Signature Standard", <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf>.

[I-D.sheffer-tls-pqc-continuity] Sheffer, Y. and T. Reddy.K, "PQC Continuity: Downgrade Protection for TLS Servers Migrating to PQC", Work in Progress, Internet-Draft, draft-sheffer-tls-pqc-continuity-01, 1 March 2026, <https://datatracker.ietf.org/doc/html/draft-sheffer-tls-pqc-continuity-01>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/rfc/rfc8017>.

[RFC9794] Driscoll, F., Parsons, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", RFC 9794, DOI 10.17487/RFC9794, June 2025, <https://www.rfc-editor.org/rfc/rfc9794>.

Rationale for a Dedicated TLS Specification (to be removed before publication)

While it might appear sufficient to allocate SignatureScheme code points for composite ML-DSA without a dedicated TLS specification, doing so would leave critical TLS-specific decisions unresolved and risk interoperability failures:

* [I-D.ietf-lamps-pq-composite-sigs] defines a context string parameter for signing and verification without mandating an empty string for TLS use; implementations could make different choices, causing signature verification failures.

* [I-D.ietf-lamps-pq-composite-sigs] defines both pure and prehashed composite ML-DSA variants; without explicit guidance, implementations could negotiate incompatible modes.

* RSASSA-PKCS1-v1_5-based composite schemes must be restricted to the signature_algorithms_cert extension and must not appear in CertificateVerify messages; this restriction cannot be inferred from code point registration alone.

* Composite ML-DSA must not be used in TLS 1.2.

* The LAMPS draft defines a larger set of composite combinations; a TLS specification is needed to define the restricted subset compatible with TLS 1.3.

Implementation Complexity Considerations (to be removed before publication)

A concern has been raised that composite signatures introduce significant API complexity for TLS implementations. This concern does not apply at the TLS layer. From the TLS perspective, a composite key is treated as a single opaque key — identical in handling to any other signature algorithm. The internal decomposition of a composite key into its ML-DSA and traditional component keys is entirely the responsibility of the underlying cryptographic library, not of the TLS implementation. A TLS implementation that supports composite ML-DSA need only handle the negotiated code points and invoke the crypto engine accordingly; the composite construction is invisible above that boundary.

Furthermore, the cryptographic library implementing composite ML-DSA can be shared across multiple protocol stacks — including IPsec, JOSE, SSH, and others — meaning the implementation effort is incurred once and benefits multiple protocols. This makes the effective per-protocol cost of supporting composite ML-DSA minimal.

Acknowledgments

Thanks to Bas Westerbaan, Alicja Kario, Ilari Liusvaara, Dan Wing, Yaron Sheffer, Samuel Lee, Eric Rescorla, and Sean Turner for the discussion and comments.

Authors' Addresses

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India
Email: kondtir@gmail.com

Timothy Hollebeek
DigiCert
Pittsburgh,
United States of America
Email: tim.hollebeek@digicert.com

John Gray
Entrust Limited
2500 Solandt Road Suite 100
Ottawa, Ontario K2K 3G5
Canada
Email: john.gray@entrust.com

Scott Fluhrer
Cisco Systems
Email: sfluhrer@cisco.com

Daniel Van Geest
CryptoNext Security
Email: daniel.vangeest@cryptonext-security.com