Limited Additional Mechanisms for PKIX and SMIME
Internet-Draft
Intended status: Informational
Expires: 14 November 2026

J. Gray
Entrust
L. Prabel
Huawei

13 May 2026

Preventing Key Reuse and Cross-Key Forgeries in Composite ML-DSA
draft-gray-lamps-compositepkc-00

Abstract

This document defines a small, backwards-compatible change to composite ML-DSA that *cryptographically binds the signature to the specific composite public key*. It does so by defining a *Public-Key Context* value (pkc) equal to a hash of the *serialized composite public key*, and by setting the composite context field to that value. This prevents *key reuse* and *cross-key forgeries* across different composite keys, while preserving the API of Composite ML-DSA. The construction introduces two helper procedures to compute pkc from either the composite private key or the composite public key.

Discussion Venues

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

Discussion of this document takes place on the Limited Additional Mechanisms for PKIX and SMIME Working Group mailing list (spasm@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/spasm/.

Source for this draft and an issue tracker can be found at https://github.com/johngray-dev/draft-gray-lamps-compositepkc.

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/.

Copyright Notice

Table of Contents

1. Introduction

Composite signature schemes (e.g., Composite ML-DSA) pre-hash the application message and prepend a *Prefix*, an algorithm-specific *Label*, and an application-provided *ctx* byte string to form the message representative M', which is then signed by each component primitive. The current composite signature construction:

M′ := Prefix || Label || len(ctx) || ctx || PH(M)

The ctx is an application context of up to 255 bytes.

While the existing design already mitigates several cross-protocol issues via Prefix and Label, and explicitly *forbids key reuse*, some deployments may still reuse component keys or attempt to combine component signatures across keys. This opens the door to *cross-key "mix-and-match" forgeries* (splicing a valid ML-DSA component from one composite with a valid traditional component from another).

This document proposes a *minimal change*: set ctx to a *hash of the composite public key*. Because the hash depends on the _exact_ public key bytes, both component signatures become bound to the same key material, preventing cross-key recombination.

2. Conventions and Definitions

This document inherits the notation of the Composite ML-DSA draft (e.g., Prefix, Label, PH, SerializePublicKey, etc.) and the conventional *KeyGen/Sign/Verify* API of a signature scheme.

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. Threat Model and Goals

Goal: prevent an adversary from taking valid component signatures produced under *different* composite keys and combining them into a valid composite signature for a target key.

Approach: bind both component signatures to the *same exact composite public key* by including pkc = H(SerializePublicKey(..)) inside M′. Because pkc changes with any bit of the key, component signatures extracted from different keys no longer verify together.

> This does *not* change the malleability properties of individual primitives (e.g., ECDSA); therefore SUF-CMA is still not claimed. It primarily removes *cross-key* splicing and reduces the impact of accidental key reuse.

4. Overview of the Construction

This construction proposes len(pkc) as the length of the len(ctx) and pkc as the ctx value; all signature computation remains as specified in the base document.

Let Hash_ctx denote the hash function chosen by the algorithm's OID (e.g., SHA-256, SHA-512, SHAKE256/64). The *Public-Key Context* is:

pkc = Hash_ctx(SerializePublicKey(mldsaPK, tradPK))

The new message representative is:

M' := Prefix || Label || len(pkc) || pkc || PH(M)

Prefix and Label are unchanged. len(pkc) is encoded as a single unsigned byte, which is sufficient because the mandated hash outputs are 64 bytes. The PH(M) is the pre-hash of the application message as in [I-D.ietf-lamps-pq-composite-sigs].

5. Public-Key Context (PKC) Routines

This draft defines two convenience routines to compute pkc from either the composite private key or the composite public key.

5.1. ComputePublicKeyContext from Private Key

Composite-ML-DSA<OID>.ComputePublicKeyContext(sk) -> pkc

Inputs:
sk: composite private key

Implicit inputs (from <OID>):
Hash_ctx: the hash function for PKC (same as the algorithm's PH unless specified otherwise)

Process:

1. (mldsaSeed, tradSK) = DeserializePrivateKey(sk)
2. (mldsaPK, mldsaSK) = ML-DSA.KeyGen_internal(mldsaSeed) // FIPS 204, seed-based expansion
4. tradPK = Trad.PublicKey(tradSK) // derive public key from private key
5. pk = SerializePublicKey(mldsaPK, tradPK)
6. pkc = Hash_ctx(pk)
7. return pkc

Notes: The seed-based ML-DSA private key representation and the ability to re-derive mldsaPK from mldsaSeed are already normative in [I-D.ietf-lamps-pq-composite-sigs]

During the signing operation, access to the public key is required. The above method suggests generating the composite public key from the composite private key by Deserializing the private key into its component keys, deriving the public component key for ML-DSA and the public component key for the traditional component, and then using the SerializePublicKey() method as defined in section 4.1 [I-D.ietf-lamps-pq-composite-sigs]. This is only one of several options, but is a non-normative, non-exhaustive list.

1. Derive or extract from private key as suggested above. Many cryptographic modules expose functionality to obtain an RSA or EC public key from the corresponding private key. For applications where such functionality does not exist, see section 10.4.1 and 10.4.2 in [I-D.ietf-lamps-pq-composite-kem] for mechanisms for extracting the public keys from private keys for RSA and ECDSA respectively. It is assumed that this is not required for Ed25519 or Ed448 since those private keys are seeds from which the public key can be obtained.

2. Fetch it from an external data source, for example from the public-key certificate corresponding to this private key.

3. If the composite signature private key is being carried within a PKCS#8 OneAsymmetricKey object, place the full composite public key within the optional OneAsymmetricKey.publicKey field (and re-encode as necessary for correctly using it in the context).

4. Use an alternate private key encoding that explicitly carries the composite public key.

5.2. ComputePublicKeyContext from Public Key

Composite-ML-DSA<OID>.ComputePublicKeyContext(pk) -> pkc

Inputs:
pk: composite public key

Implicit inputs (from <OID>):
Hash_ctx

Process:
1. pkc = Hash_ctx(pk)
2. return pkc

6. Algorithms

Both Composite-ML-DSA<OID>.Sign(sk, M, ctx) and Composite-ML-DSA<OID>.Verify(pk, M, s, ctx) remain the same. The only difference is that pkc is passed in as the ctx value.

The signature interface remains the same.

Note: The application specific ctx argument is *ignored* with this current design. To keep that property, a future version of this specification could set pkc = HASH (ctx || publickey).

7. Serialization and ASN.1 Usage

This document *does not* change the composite public/private key or signature *serialization formats* from [I-D.ietf-lamps-pq-composite-sigs] and signatures remain concatenations of the component encodings. It also does not change DER wrapping in SPKI/PKCS#8.

Because wire compatibility requires peers to know whether ctx is application-set or PKC-bound, this document could have registered *new algorithm identifiers* for each PKC-bound combination. However, that is not within the scope of this document. This is meant for specific application context use-cases where the preventing key reuse is a desired security property. For example, applications which choose to profile a set of composite signatures could choose to also adopt the use of this context.

8. Security Considerations

Key Reuse: [I-D.ietf-lamps-pq-composite-sigs] strictly forbids reusing component keys. Binding ctx to pkc provides a cryptographic backstop: even if component keys were (improperly) reused, cross-key splicing will fail because pkc differs for each public key instance.

Non-separability: [I-D.ietf-lamps-pq-composite-sigs] achieved Weak Non-Separability (WNS) and a limited form of SNS for ML-DSA via the mldsa_ctx=Label. PKC-binding additionally prevents forming (mldsaSig1, tradSig2) under different keys, because both signatures are now bounded to the same pkc. This does *not* fix primitive-level malleability (e.g., ECDSA) and therefore does not claim SUF-CMA. However, for algorithms like EdDSA or Ed448 which are SUF secure, this property should remain.

Prefix: Existing *Prefix* and *Label* remain unchanged; deployments that implemented the optional Prefix in traditional verifiers can keep it as is.

Hash Choices: Hash_ctx MUST be the algorithm’ s registered pre-hash function (e.g., SHA-256, SHA-512, SHAKE256/64). This keeps implementation complexity minimal and ensures digest sizes fit within the ctx length field.

Privacy: pkc reveals nothing beyond what the public key already reveals; it is a hash of public data.

9. Implementation Considerations

Signer Access to pk: The signer computes pkc either by deriving compositePK from compositeSK, or by keeping a cached copy of compositepk alongside compositesk.

Interoperability: Because M’ changes when this context type is used, peers MUST know that this context will be used. One way to achieve this is for application specific use cases to specify this context type as part of the usage. For example, if an application using composite signatures desired this security property, it could make use of the public key binding in the context mandatory.

10. IANA Considerations

None

11. References

11.1. Normative References

[FIPS.204] National Institute of Standards and Technology (NIST), "Module-Lattice-Based Digital Signature Standard", FIPS PUB 204, August 2024, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf>.

[I-D.ietf-lamps-pq-composite-kem] Ounsworth, M., Gray, J., Pala, M., Klauner, J., and S. Fluhrer, "Composite ML-KEM for use in X.509 Public Key Infrastructure", Work in Progress, Internet-Draft, draft-ietf-lamps-pq-composite-kem-14, 27 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-pq-composite-kem-14>.

[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>.

11.2. Informative References

[I-D.draft-ietf-pquip-hybrid-signature-spectrums-07] 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>.

Appendix A. Acknowledgments

Thanks to the Composite ML-DSA authors and LAMPS WG for the existing design and analyses of pre-hashing, non-separability, and key-reuse risks which this document builds upon. Thanks to Daniel Van Geest for his feedback on this document.

Authors' Addresses

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

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