Stream:	Internet Engineering Task Force (IETF)
RFC:	10024
Category:	Standards Track
Published:	August 2026
ISSN:	2070-1721
Authors:	K. Kwiatkowski	P. Kampanakis	B. E. Westerbaan	D. Stebila
---	---	---	---	---
	PQShield	*AWS*	*Cloudflare*	*University of Waterloo*

RFC 10024
Post-Quantum Traditional (PQ/T) Hybrid Key Agreement Mechanisms for TLS 1.3

Abstract

This document defines three hybrid key agreement mechanisms for TLS 1.3 -- X25519MLKEM768, SecP256r1MLKEM768, and SecP384r1MLKEM1024 -- that combine the post-quantum ML-KEM (Module-Lattice-Based Key Encapsulation Mechanism) with an ECDHE (Ephemeral Elliptic Curve Diffie-Hellman) exchange.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc10024.

Copyright Notice

Table of Contents

1. Introduction

ML-KEM is a key encapsulation mechanism (KEM) defined in [NIST-FIPS-203]. It is designed to withstand cryptanalytic attacks from quantum computers.

[RFC9954] defines a framework for combining traditional key exchanges with next-generation key exchange in TLS 1.3. The goal of this approach is to provide security against both classical and quantum adversaries while maintaining compatibility with existing infrastructure and protocols.

This document applies the framework in [RFC9954] to ML-KEM and specifies code points for the hybrid groups.

2. Motivation

This document introduces three new supported groups for Post-Quantum Traditional (PQ/T) hybrid key agreements [RFC9794] in TLS 1.3 -- X25519MLKEM768, SecP256r1MLKEM768, and SecP384r1MLKEM1024 -- that combine ML-KEM with Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) in the manner described in [RFC9954]. Any of the hybrid groups specified in this document may be implemented in a FIPS-approved way as discussed in Section 5.

- The first group uses X25519 [RFC7748], is widely deployed, and often serves as the most practical choice for a single PQ/T hybrid combiner [RFC9794] in TLS 1.3.
- The second group uses secp256r1 (NIST P-256) [NIST-FIPS-186]. This group supports use cases that require both shared secrets to be generated by FIPS-approved mechanisms.
- The third group uses secp384r1 (NIST P-384) [NIST-FIPS-186]. This group is intended for high-security environments that require FIPS-approved mechanisms with an increased security margin.

Key establishment using NIST curves is outlined in Section 6.1.2.2 of [NIST-SP-800-56A].

3. Terminology

[RFC9954] defines "traditional" algorithms as those that are already widely adopted and "next-generation" algorithms as those that are not yet widely adopted, such as post-quantum algorithms. In this document, ECDHE using Curve25519, P-256, or P-384 is considered traditional, while ML-KEM is considered next-generation.

[RFC9954] also defines a "hybrid" key exchange as the simultaneous use of multiple key exchange algorithms, with their outputs combined to provide security as long as at least one of the component algorithms remains secure, even if the others are compromised. This document uses the term "hybrid" with the same meaning.

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.

4. Negotiated Groups

4.1. Client Share

When the X25519MLKEM768 group is negotiated, the client's key_exchange value is the concatenation of the client's ML-KEM-768 encapsulation key and the client's X25519 ephemeral share. The size of the client share is 1216 bytes (1184 bytes for the ML-KEM part and 32 bytes for X25519).

> Note: The group name X25519MLKEM768 does not adhere to the naming convention outlined in Section 3.2 of [RFC9954]. Specifically, the order of shares in the concatenation has been reversed. This is due to historical reasons.

When the SecP256r1MLKEM768 group is negotiated, the client's key_exchange value is the concatenation of the secp256r1 ephemeral share and ML-KEM-768 encapsulation key. The ECDHE share is the serialized value of the uncompressed ECDHE point representation as defined in Section 4.3.8.2 of [RFC9846]. The size of the client share is 1249 bytes (65 bytes for the secp256r1 part and 1184 bytes for ML-KEM).

When the SecP384r1MLKEM1024 group is negotiated, the client's key_exchange value is the concatenation of the secp384r1 ephemeral share and the ML-KEM-1024 encapsulation key. The ECDHE share is the serialized value of the uncompressed ECDHE point representation as defined in Section 4.3.8.2 of [RFC9846]. The size of the client share is 1665 bytes (97 bytes for the secp384r1 part and 1568 for ML-KEM).

4.2. Server Share

When the X25519MLKEM768 group is negotiated, the server's key_exchange value is the concatenation of an ML-KEM ciphertext returned from encapsulation to the client's encapsulation key and the server's ephemeral X25519 share. The size of the server share is 1120 bytes (1088 bytes for the ML-KEM part and 32 bytes for X25519).

When the SecP256r1MLKEM768 group is negotiated, the server's key_exchange value is the concatenation of the server's ephemeral secp256r1 share encoded in the same way as the client share and an ML-KEM ciphertext returned from encapsulation to the client's encapsulation key. The size of the server share is 1153 bytes (1088 bytes for the ML-KEM part and 65 bytes for secp256r1).

When the SecP384r1MLKEM1024 group is negotiated, the server's key_exchange value is the concatenation of the server's ephemeral secp384r1 share encoded in the same way as the client share and an ML-KEM ciphertext returned from encapsulation to the client's encapsulation key. The size of the server share is 1665 bytes (1568 bytes for the ML-KEM part and 97 bytes for secp384r1).

For all groups, the server **MUST** perform the encapsulation key check described in Section 7.2 of [NIST-FIPS-203] on the client's encapsulation key and abort with an illegal_parameter alert if it fails.

For all groups, the client **MUST** check if the ciphertext length matches the selected group and abort with an illegal_parameter alert if it fails. If ML-KEM decapsulation fails for any other reason, the connection **MUST** be aborted with an internal_error alert.

For all groups, both client and server **MUST** process the ECDHE part as described in Section 4.3.8.2 of [RFC9846], including all validity checks, and abort with an illegal_parameter alert if it fails.

4.3. Shared Secret

For X25519MLKEM768, the shared secret is the concatenation of the ML-KEM shared secret and the X25519 shared secret. The shared secret is 64 bytes (32 bytes for each part).

For SecP256r1MLKEM768, the shared secret is the concatenation of the ECDHE and ML-KEM shared secrets. The ECDHE shared secret is the x-coordinate of the ECDHE shared secret elliptic curve point represented as an octet string as defined in Section 7.4.2 of [RFC9846]. The size of the shared secret is 64 bytes (32 bytes for each part).

For SecP384r1MLKEM1024, the shared secret is the concatenation of the ECDHE and ML-KEM shared secrets. The ECDHE shared secret is the x-coordinate of the ECDHE shared secret elliptic curve point represented as an octet string as defined in Section 7.4.2 of [RFC9846]. The size of the shared secret is 80 bytes (48 bytes for the ECDHE part and 32 bytes for the ML-KEM part).

For all groups, both client and server **MUST** calculate the ECDHE part of the shared secret as described in Section 7.4.2 of [RFC9846], including the all-zero shared secret check for X25519, and abort the connection with an illegal_parameter alert if it fails.

5. Regulatory Context

This section provides informal notes on how the hybrid key agreement mechanisms defined in this document relate to existing NIST guidance on key derivation and hybrid key establishment.

- **FIPS-compliance**. All groups defined in this document permit FIPS-approved key derivation as per [NIST-SP-800-56C] and [NIST-SP-800-135]. NIST Special Publication 800-56Cr2 [NIST-SP-800-56C] approves the usage of the HMAC-based Key Derivation Function (HKDF) [RFC5869] with two distinct shared secrets, with the condition that the first one is computed by a FIPS-approved key-establishment scheme. FIPS also requires a certified implementation of the scheme, which will remain more ubiquitous for secp256r1 in the coming years. For this reason, the ML-KEM shared secret is placed first in X25519MLKEM768, while the ECDHE shared secret is placed first in SecP256r1MLKEM768 and SecP384r1MLKEM1024. This means that for SecP256r1MLKEM768 and SecP384r1MLKEM1024, the ECDHE implementation must be certified, whereas the ML-KEM implementation does not require certification. In contrast, for X25519MLKEM768, the ML-KEM implementation must be certified.
- **SP800-227 compliance**. NIST Special Publication 800-227 [NIST-SP-800-227] provides general guidance on the design and use of key encapsulation mechanisms, including hybrid constructions. The key agreements defined in this document follow the principles described in Section 4.6 of [NIST-SP-800-227], which discusses the combination of post-quantum and classical key-establishment schemes and the use of approved key combiners. In particular, the shared-secret concatenation and HKDF-based derivation used by TLS 1.3 are consistent with the composite-KEM constructions and key-combiner recommendations outlined in Sections 4.6.1 and 4.6.2 of [NIST-SP-800-227]. Section 4.6.3 of [NIST-SP-800-227] further provides relevant security considerations for hybrid KEM designs underlying the approach used in this document.

6. Security Considerations

The same security considerations as those described in [RFC9954] apply to the approach used by this document. The security analysis relies crucially on the TLS 1.3 message transcript, and one cannot assume a similar hybridization is secure in other protocols.

[NIST-SP-800-227] includes guidelines and requirements for implementations on using KEMs securely. Implementers are encouraged to use implementations resistant to side-channel attacks, especially those that can be applied by remote attackers.

All groups defined in this document use and generate fixed-length public keys, ciphertexts, and shared secrets, which complies with the requirements described in Section 6 of [RFC9954].

During ML-KEM encapsulation, encapsulation randomness `m` is drawn from a random bit generator and encrypted (see [NIST-FIPS-203], Algorithms 17 and 20); the client, which holds the decapsulation key, then recovers `m` exactly during decapsulation (see [NIST-FIPS-203], Algorithm 18). Consequently, any information `m` carries about the generator's other outputs is also exposed to the client.

The disclosure of the output(s) of an insecure random number generator (RNG) when used in TLS can be used in an attack to compromise the state of the insecure RNG itself as described in [DUALECTLS]. The encapsulation randomness `m` in ML-KEM is an additional place where RNG output is disclosed to an active attacker. Implementers should follow the RBG guidance in [NIST-FIPS-203] and the random number generation guidance in Appendix C.1 of [RFC9846]. Implementers can choose to implement mechanisms from [RFC8937] for additional protection across sessions.

In contrast, the ECDH ephemeral scalars taken from the RNG are never directly disclosed to the peer. However, any passive observer with access to a cryptographically relevant quantum computer (CRQC) can recover the scalar, which is derived directly from RNG output. Regardless, ephemeral scalars should always be generated using a cryptographically secure RNG: for secp256r1 and secp384r1 as required by [NIST-SP-800-56A], and for X25519 as described in [RFC7748]; the guidance in Appendix C.1 of [RFC9846] applies here as well.

If the same insecure RNG is used by both algorithms, then a disclosure of state by one of the algorithms will also affect the security of the other algorithm.

7. IANA Considerations

Per this document, IANA has registered three new entries in the "TLS Supported Groups" registry, according to the procedures in Section 6 of [RFC9847]. These identifiers are to be used with the final version of ML-KEM ratified by NIST, which is specified in [NIST-FIPS-203].

7.1. X25519MLKEM768

Value: 4588 (0x11EC)

Description: X25519MLKEM768

DTLS-OK: Y

Recommended: Y

Reference: RFC 10024

Comment: Combining X25519 ECDH with ML-KEM-768

7.2. SecP256r1MLKEM768

Value: 4587 (0x11EB)

Description: SecP256r1MLKEM768

DTLS-OK: Y

Recommended: N

Reference: RFC 10024

Comment: Combining secp256r1 ECDH with ML-KEM-768

7.3. SecP384r1MLKEM1024

Value: 4589 (0x11ED)

Description: SecP384r1MLKEM1024

DTLS-OK: Y

Recommended: N

Reference: RFC 10024

Comment: Combining secp384r1 ECDH with ML-KEM-1024

7.4. Obsoleted Supported Groups

Experimental code points for pre-standard versions of Kyber768 were added to the "TLS Supported Groups" registry as X25519Kyber768Draft00 (25497) and SecP256r1Kyber768Draft00 (25498). This document obsoletes these entries. For both entries, IANA has modified the Recommended field to 'D', added this document as a reference, and updated the Comment field to "Pre-standards version of Kyber768. Obsoleted by RFC 10024."

8. References

8.1. Normative References

[NIST-FIPS-186] NIST, "Digital Signature Standard (DSS)", NIST FIPS 186-5, DOI 10.6028/NIST.FIPS.186-5, February 2023, <https://doi.org/10.6028/NIST.FIPS.186-5>.

[NIST-FIPS-203] NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", NIST FIPS 203, DOI 10.6028/NIST.FIPS.203, August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[NIST-SP-800-56C] Barker, E., Chen, L., and R. Davis, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes", National Institute of Standards and Technology, NIST SP 800-56Cr2, DOI 10.6028/nist.sp.800-56cr2, August 2020, <https://doi.org/10.6028/nist.sp.800-56cr2>.

[NIST-SP-800-135] Dang, Q., "Recommendation for Existing Application-Specific Key Derivation Functions", National Institute of Standards and Technology, NIST SP 800-135r1, DOI 10.6028/nist.sp.800-135r1, December 2011, <https://doi.org/10.6028/nist.sp.800-135r1>.

[NIST-SP-800-227] Alagic, G., Barker, E., Chen, L., Moody, D., Robinson, A., Silberg, H., and N. Waller, "Recommendations for Key-Encapsulation Mechanisms", National Institute of Standards and Technology, NIST SP 800-227, DOI 10.6028/nist.sp.800-227, September 2025, <https://doi.org/10.6028/nist.sp.800-227>.

[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>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[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>.

[RFC9846] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 9846, DOI 10.17487/RFC9846, July 2026, <https://www.rfc-editor.org/info/rfc9846>.

[RFC9954] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid Key Exchange in TLS 1.3", RFC 9954, DOI 10.17487/RFC9954, July 2026, <https://www.rfc-editor.org/info/rfc9954>.

8.2. Informative References

[DUALECTLS]

Checkoway, S., Fredrikson, M., Niederhagen, R., Everspaugh, A., Green, M., Lange, T., Ristenpart, T., Bernstein, D. J., Maskiewicz, J., and H. Shacham, "On the Practical Exploitability of Dual EC in TLS Implementations", 23rd USENIX Security Symposium (USENIX Security 14), 2014, <https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-checkoway.pdf>.

[NIST-SP-800-56A] Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", National Institute of Standards and Technology, NIST SP 800-56Ar3, DOI 10.6028/nist.sp.800-56ar3, April 2018, <https://doi.org/10.6028/nist.sp.800-56ar3>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/info/rfc8937>.

[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/info/rfc9794>.

[RFC9847] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 9847, DOI 10.17487/RFC9847, December 2025, <https://www.rfc-editor.org/info/rfc9847>.

Authors' Addresses

Krzysztof Kwiatkowski
PQShield
Email: kris@amongbytes.com

Panos Kampanakis
AWS
Email: kpanos@amazon.com

Bas Westerbaan
Cloudflare
Email: bas@cloudflare.com

Douglas Stebila
University of Waterloo
Email: dstebila@uwaterloo.ca