Stream:	Internet Engineering Task Force (IETF)		
RFC:	9954		
Category:	Informational		
Published:	April 2026		
ISSN:	2070-1721		
Authors:	D. Stebila	S. Fluhrer	S. Gueron
	University of Waterloo	*Cisco Systems*	*U. Haifa & Meta*

RFC 9954
Hybrid Key Exchange in TLS 1.3

Abstract

Hybrid key exchange refers to using multiple key exchange algorithms simultaneously and combining the result with the goal of providing security even if a way is found to defeat the encryption for all but one of the component algorithms. It is motivated by the transition to post-quantum cryptography. This document provides a construction for hybrid key exchange in the Transport Layer Security (TLS) protocol version 1.3.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

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). Not all documents approved by the IESG are candidates for any level of Internet Standard; see 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/rfc9954.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Terminology 3
1.2. Motivation for Use of Hybrid Key Exchange 4
1.3. Scope 4
1.4. Goals 5
2. Key Encapsulation Mechanisms 6
3. Construction for Hybrid Key Exchange 7
3.1. Negotiation 7
3.2. Transmitting Public Keys and Ciphertexts 7
3.3. Shared Secret Calculation 9
4. Discussion 10
5. IANA Considerations 11
6. Security Considerations 11
7. References 12
7.1. Normative References 12
7.2. Informative References 12
Appendix A. Related Work 16
Acknowledgements 17
Authors' Addresses 17

1. Introduction

This document gives a construction for hybrid key exchange in TLS 1.3. The overall design approach is a simple, "concatenation"-based approach: Each hybrid key exchange combination should be viewed as a single new key exchange method, negotiated and transmitted using the existing TLS 1.3 mechanisms.

This document does not propose specific post-quantum mechanisms; see Section 1.3 for more on the scope of this document.

1.1. 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.

For the purposes of this document, it is helpful to divide cryptographic algorithms into two classes:

- "Traditional" algorithms: Algorithms that are widely deployed today but may be deprecated in the future. In the context of TLS 1.3, examples of traditional key exchange algorithms include Elliptic Curve Diffie-Hellman (ECDH) using secp256r1 or x25519 or finite field Diffie-Hellman.
- "Next-generation" (or "next-gen") algorithms: Algorithms that are not yet widely deployed but may eventually be widely deployed. An additional facet of these algorithms may be that the cryptographic community has less confidence in their security due to them being relatively new or less studied. This includes "post-quantum" algorithms.

In this context, "hybrid" key exchange means the use of two (or more) key exchange algorithms based on different cryptographic assumptions, e.g., one traditional algorithm and one next-generation algorithm, with the purpose of the final session key being secure as long as at least one of the component key exchange algorithms remains unbroken. When one of the algorithms is traditional and one is post-quantum, this is a Post-Quantum Traditional Hybrid Scheme [PQUIP-TERM]; while this is the initial use case for this document, the document is not limited to this case. This document uses the term "component" algorithms to refer to the algorithms combined in a hybrid key exchange.

Some researchers prefer the term "composite" to refer to the use of multiple algorithms to distinguish from "hybrid public key encryption", in which a key encapsulation mechanism and data encapsulation mechanism are combined to create public key encryption.

It is intended that the component algorithms within a hybrid key exchange are to be performed, that is, negotiated and transmitted, within the TLS 1.3 handshake. Any out-of-band method of exchanging keying material is considered out-of-scope.

The primary motivation of this document is preparing for post-quantum algorithms. However, it is possible that public key cryptography based on alternative mathematical constructions will be desired to mitigate risks independent of the advent of a quantum computer, for example, because of a cryptanalytic breakthrough. As such, this document opts for the more generic term "next-generation" algorithms rather than exclusively "post-quantum" algorithms.

Note that TLS 1.3 uses the term "groups" to refer to key exchange algorithms -- for example, the `supported_groups` extension -- since all key exchange algorithms in TLS 1.3 are Diffie-Hellman-based. As a result, some parts of this document will refer to data structures or messages with the term "group" in them despite using a key exchange algorithm that is neither Diffie-Hellman-based nor a group.

1.2. Motivation for Use of Hybrid Key Exchange

A hybrid key exchange algorithm allows early adopters eager for post-quantum security to have the potential of post-quantum security (possibly from a less-well-studied algorithm) while still retaining at least the security currently offered by traditional algorithms. They may even need to retain traditional algorithms due to regulatory constraints, for example, US National Institute of Standards and Technology (NIST) FIPS compliance.

Ideally, one would not use hybrid key exchange: One would have confidence in a single algorithm and parameterization that will stand the test of time. However, this may not be the case in the face of quantum computers and cryptanalytic advances more generally.

Many (though not all) post-quantum algorithms currently under consideration are relatively new; they have not been subject to the same depth of study as RSA and finite field or elliptic curve Diffie-Hellman; thus, the security community does not necessarily have as much confidence in their fundamental security or the concrete security level of specific parameterizations.

Moreover, it is possible that after next-generation algorithms are defined, and for a period of time thereafter, conservative users may not have full confidence in some algorithms.

Some users may want to accelerate adoption of post-quantum cryptography due to the threat of retroactive decryption (also known as "harvest-now-decrypt-later"): If a cryptographic assumption is broken due to the advent of a quantum computer or some other cryptanalytic breakthrough, confidentiality of information can be broken retroactively by any adversary who has passively recorded handshakes and encrypted communications. Hybrid key exchange enables potential security against retroactive decryption while not fully abandoning traditional cryptosystems.

As such, there may be users for whom hybrid key exchange is an appropriate step prior to an eventual transition to next-generation algorithms. Users should consider the confidence they have in each hybrid component to assess that the hybrid system meets the desired motivation.

1.3. Scope

This document focuses on hybrid ephemeral key exchange in TLS 1.3 [TLS13]. It intentionally does not address:

- Selecting which next-generation algorithms to use in TLS 1.3 or algorithm identifiers or encoding mechanisms for next-generation algorithms.

- Authentication using next-generation algorithms. While quantum computers could retroactively decrypt previous sessions, session authentication cannot be retroactively broken.

1.4. Goals

The primary goal of a hybrid key exchange mechanism is to facilitate the establishment of a shared secret that remains secure as long as one of the component key exchange mechanisms remains unbroken.

In addition to the primary cryptographic goal, there may be several additional goals in the context of TLS 1.3:

- **Backwards compatibility**: Clients and servers who are "hybrid-aware", i.e., compliant with whatever hybrid key exchange standard is developed for TLS, should remain compatible with endpoints and middleboxes that are not hybrid-aware. The three scenarios to consider are:
  1. Hybrid-aware client, hybrid-aware server: These parties should establish a hybrid shared secret.
  2. Hybrid-aware client, non-hybrid-aware server: These parties should establish a non-hybrid shared secret (assuming the hybrid-aware client is willing to downgrade to non-hybrid-only).
  3. Non-hybrid-aware client, hybrid-aware server: These parties should establish a non-hybrid shared secret (assuming the hybrid-aware server is willing to downgrade to non-hybrid-only).

  Ideally, backwards compatibility should be achieved without extra round trips and without sending duplicate information; see below.
- **High performance**: Use of hybrid key exchange should not be prohibitively expensive in terms of computational performance. In general, this will depend on the performance characteristics of the specific cryptographic algorithms used and, as such, is outside the scope of this document. See [PST] for preliminary results about performance characteristics.
- **Low latency**: Use of hybrid key exchange should not substantially increase the latency experienced to establish a connection. Factors affecting this may include the following:
  - The computational performance characteristics of the specific algorithms used. See above.
  - The size of messages to be transmitted. Public key and ciphertext sizes for post-quantum algorithms range from hundreds of bytes to over one hundred kilobytes, so this impact can be substantial. See [PST] for preliminary results in a laboratory setting and [LANGLEY] for preliminary results on more realistic networks.
  - Additional round trips added to the protocol. See below.
- **No extra round trips**: Attempting to negotiate hybrid key exchange should not lead to extra round trips in any of the three hybrid-aware/non-hybrid-aware scenarios listed above.
- **Minimal duplicate information**: Attempting to negotiate hybrid key exchange should not mean having to send multiple public keys of the same type.

The tolerance for lower performance and increased latency due to use of hybrid key exchange will depend on the context and use case of the systems and the network involved.

2. Key Encapsulation Mechanisms

This document models key agreement as key encapsulation mechanisms (KEMs), which consist of three algorithms:

- `KeyGen() -> (pk, sk)`: A probabilistic key generation algorithm, which generates a public key `pk` and a secret key `sk`.
- `Encaps(pk) -> (ct, ss)`: A probabilistic encapsulation algorithm, which takes as input a public key `pk` and outputs a ciphertext `ct` and shared secret `ss`.
- `Decaps(sk, ct) -> ss`: A decapsulation algorithm, which takes as input a secret key `sk` and ciphertext `ct` and outputs a shared secret `ss` or, in some cases, a distinguished error value.

The main security property for KEMs is indistinguishability under adaptive chosen ciphertext attack (IND-CCA2), which means that shared secret values should be indistinguishable from random strings even given the ability to have other arbitrary ciphertexts decapsulated. IND-CCA2 corresponds to security against an active attacker, and the public key and secret key pair can be treated as a long-term key or reused (see, for example, [KATZ] or [HHK] for definitions of IND-CCA2 and IND-CPA security). A common design pattern for obtaining security under key reuse is to apply the Fujisaki-Okamoto (FO) transform [FO] or a variant thereof [HHK].

A weaker security notion is indistinguishability under chosen plaintext attack (IND-CPA), which means that the shared secret values should be indistinguishable from random strings given a copy of the public key. IND-CPA roughly corresponds to security against a passive attacker and sometimes corresponds to one-time key exchange.

Key exchange in TLS 1.3 is phrased in terms of Diffie-Hellman key exchange in a group. DH key exchange can be modeled as a KEM, with `KeyGen` corresponding to selecting an exponent `x` as the secret key and computing the public key `g^x`, encapsulation corresponding to selecting an exponent `y` and computing the ciphertext `g^y` and the shared secret `g^(xy)`, and decapsulation as computing the shared secret `g^(xy)`. See [HPKE] for more details of such Diffie-Hellman-based key encapsulation mechanisms. Diffie-Hellman key exchange, when viewed as a KEM, does not formally satisfy IND-CCA2 security but is still safe to use for ephemeral key exchange in TLS 1.3; see, for example, [DOWLING].

TLS 1.3 does not require that ephemeral public keys be used only in a single key exchange session; some implementations may reuse them, at the cost of limited forward secrecy. As a result, any KEM used in the manner described in this document **MUST** explicitly be designed to be secure in the event that the public key is reused. Finite field and elliptic curve Diffie-Hellman key exchange methods used in TLS 1.3 satisfy this criteria. For generic KEMs, this means satisfying IND-CCA2 security or having a transform like the Fujisaki-Okamoto transform [FO] [HHK] applied. While it is recommended that implementations avoid reuse of KEM public keys,

implementations that do reuse KEM public keys **MUST** ensure that the number of reuses of a KEM public key abides by any bounds in the specification of the KEM or subsequent security analyses. Implementations **MUST NOT** reuse randomness in the generation of KEM ciphertexts.

3. Construction for Hybrid Key Exchange

3.1. Negotiation

Each particular combination of algorithms in a hybrid key exchange will be represented as a `NamedGroup` and sent in the `supported_groups` extension. No internal structure or grammar is implied or required in the value of the identifier; they are simply opaque identifiers.

Each value representing a hybrid key exchange will correspond to an ordered pair of two or more algorithms. (Note that this is independent from future documents standardizing solely post-quantum key exchange methods, which would have to be assigned their own identifier.)

3.2. Transmitting Public Keys and Ciphertexts

This document takes the relatively simple "concatenation approach": The messages from the two or more algorithms being hybridized will be concatenated together and transmitted as a single value to avoid having to change existing data structures. The values are directly concatenated, without any additional encoding or length fields; the representation and length of elements **MUST** be fixed once the algorithm is fixed.

Recall that, in TLS 1.3 ([TLS13], Section 4.2.8), a KEM public key or KEM ciphertext is represented as a `KeyShareEntry`:

```
struct {
    NamedGroup group;
    opaque key_exchange<1..2^16-1>;
} KeyShareEntry;
```

These are transmitted in the `extension_data` fields of `KeyShareClientHello` and `KeyShareServerHello` extensions:

```
struct {
    KeyShareEntry client_shares<0..2^16-1>;
} KeyShareClientHello;

struct {
    KeyShareEntry server_share;
} KeyShareServerHello;
```

The client's shares are listed in descending order of client preference; the server selects one algorithm and sends its corresponding share.

For a hybrid key exchange, the `key_exchange` field of a `KeyShareEntry` is the concatenation of the `key_exchange` field for each of the constituent algorithms. The order of shares in the concatenation **MUST** be the same as the order of algorithms indicated in the definition of the `NamedGroup`.

For the client's share, the `key_exchange` value contains the concatenation of the `pk` outputs of the corresponding KEMs' `KeyGen` algorithms if that algorithm corresponds to a KEM or the (EC)DH ephemeral key share if that algorithm corresponds to an (EC)DH group. For the server's share, the `key_exchange` value contains concatenation of the `ct` outputs of the corresponding KEMs' `Encaps` algorithms if that algorithm corresponds to a KEM or the (EC)DH ephemeral key share if that algorithm corresponds to an (EC)DH group.

Section 4.2.8 of [TLS13] requires that "The key_exchange values for each KeyShareEntry **MUST** be generated independently." In the context of this document, the same algorithm may appear in multiple named groups; thus, this document relaxes the above requirement to allow the same key_exchange value for the same algorithm to be reused in multiple KeyShareEntry records sent within the same `ClientHello`. However, key_exchange values for different algorithms **MUST** be generated independently. Explicitly, if the `NamedGroup` is the hybrid key exchange `MyECDHMyPQKEM`, the `KeyShareEntry.key_exchange` values **MUST** be generated in one of the following two ways:

Fully independently:

```
MyECDHMyPQKEM.KeyGen() = (MyECDH.KeyGen(), MyPQKEM.KeyGen())

KeyShareClientHello {
    KeyShareEntry {
        NamedGroup: 'MyECDH',
        key_exchange: MyECDH.KeyGen()
    },
    KeyShareEntry {
        NamedGroup: 'MyPQKEM',
        key_exchange: MyPQKEM.KeyGen()
    },
    KeyShareEntry {
        NamedGroup: 'MyECDHMyPQKEM',
        key_exchange: MyECDHMyPQKEM.KeyGen()
    },
}
```

Reusing key_exchange values of the same component algorithm within the same `ClientHello`:

```
myecdh_key_share = MyECDH.KeyGen()
mypqkem_key_share = MyPQKEM.KeyGen()
myecdh_mypqkem_key_share = (myecdh_key_share, mypqkem_key_share)

KeyShareClientHello {
    KeyShareEntry {
        NamedGroup: 'MyECDH',
        key_exchange: myecdh_key_share
    },
    KeyShareEntry {
        NamedGroup: 'MyPQKEM',
        key_exchange: mypqkem_key_share
    },
    KeyShareEntry {
        NamedGroup: 'MyECDHMyPQKEM',
        key_exchange: myecdh_mypqkem_key_share
    },
}
```

3.3. Shared Secret Calculation

Here, this document also takes a simple "concatenation approach": The two shared secrets are concatenated together and used as the shared secret in the existing TLS 1.3 key schedule. Again, this document does not add any additional structure (length fields) in the concatenation procedure; for both the traditional groups and post quantum KEMs, the shared secret output length is fixed for a specific elliptic curve or parameter set.

In other words, if the `NamedGroup` is `MyECDHMyPQKEM`, the shared secret is calculated as:

```
concatenated_shared_secret = MyECDH.shared_secret || MyPQKEM.shared_secret
```

and inserted into the TLS 1.3 key schedule in place of the (EC)DHE shared secret, as shown in Figure 1.

```
                                 0
                                 |
                                 v
                   PSK ->  HKDF-Extract = Early Secret
                                 |
                                 +-----> Derive-Secret(...)
                                 +-----> Derive-Secret(...)
                                 +-----> Derive-Secret(...)
                                 |
                                 v
                           Derive-Secret(., "derived", "")
                                 |
                                 v
concatenated_shared_secret -> HKDF-Extract = Handshake Secret
^^^^^^^^^^^^^^^^^^^^^^^^^^       |
                                 +-----> Derive-Secret(...)
                                 +-----> Derive-Secret(...)
                                 |
                                 v
                           Derive-Secret(., "derived", "")
                                 |
                                 v
                      0 -> HKDF-Extract = Master Secret
                                 |
                                 +-----> Derive-Secret(...)
                                 +-----> Derive-Secret(...)
                                 +-----> Derive-Secret(...)
                                 +-----> Derive-Secret(...)
```

Figure 1: Key Schedule for Hybrid Key Exchange

FIPS compliance of shared secret concatenation. The US National Institute of Standards and Technology (NIST) documents [NIST-SP-800-56C] and [NIST-SP-800-135] give recommendations for key derivation methods in key exchange protocols. Some hybrid combinations may combine the shared secret from a NIST-approved algorithm (e.g., ECDH using the nistp256/secp256r1 curve) with a shared secret from a non-approved algorithm (e.g., post-quantum). [NIST-SP-800-56C] lists simple concatenation as an approved method for generation of a hybrid shared secret in which one of the constituent shared secrets is from an approved method.

4. Discussion

Larger public keys and/or ciphertexts. The `key_exchange` field in the `KeyShareEntry` struct in Section 3.2 limits public keys and ciphertexts to 2^{16}-1 bytes. Some post-quantum KEMs have larger public keys and/or ciphertexts; for example, Classic McEliece's smallest parameter set has a public key size of 261,120 bytes. However, all defined parameter sets for the Module-Lattice-Based Key Encapsulation Mechanism (ML-KEM) [NIST-FIPS-203] have public keys and ciphertexts that fall within the TLS constraints (although this may result in ClientHello messages larger than a single packet).

Duplication of key shares. Concatenation of public keys in the `key_exchange` field in the `KeyShareEntry` struct as described in Section 3.2 can result in sending duplicate key shares. For example, if a client wants to offer support for two combinations, say "SecP256r1MLKEM768" and "X25519MLKEM768" [ECDHE-MLKEM], it would end up sending two ML-KEM-768 public keys, since the `KeyShareEntry` for each combination contains its own copy of an ML-KEM-768 key. This duplication may be more problematic for post-quantum algorithms that have larger public keys. On the other hand, if the client wants to offer, for example, "SecP256r1MLKEM768" and "secp256r1" (for backwards compatibility), there is relatively little duplicated data (as the secp256r1 keys are comparatively small).

5. IANA Considerations

IANA has added this document as a reference for the "TLS Supported Groups" registry [IANA-TLS].

For hybrid combinations defined per this document, IANA will assign identifiers in a range that is distinct from the Finite Field Groups range. In addition, the "Recommended" column **SHOULD** be "N", and the "DTLS-OK" column **SHOULD** be "Y".

6. Security Considerations

The shared secrets computed in the hybrid key exchange should be computed in a way that achieves the "hybrid" property: The resulting secret is secure as long as at least one of the component key exchange algorithms is unbroken. See [GIACON] and [BINDEL] for an investigation of these issues. Under the assumption that shared secrets are fixed length once the combination is fixed, the construction in Section 3.3 corresponds to the dual-PRF combiner of [BINDEL], which is shown to preserve security under the assumption that the hash function is a dual-PRF.

As noted in Section 2, KEMs used in the manner described in this document **MUST** explicitly be designed to be secure in the event that the public key is reused, such as achieving IND-CCA2 security or having a transform like the Fujisaki-Okamoto transform applied. ML-KEM has such security properties. However, some other post-quantum KEMs designed to be IND-CPA-secure (i.e., without countermeasures such as the FO transform) are completely insecure under public key reuse; for example, some lattice-based IND-CPA-secure KEMs are vulnerable to attacks that recover the private key after just a few thousand samples [FLUHRER].

Public keys, ciphertexts, and secrets should be constant length. This document assumes that the length of each public key, ciphertext, and shared secret is fixed once the algorithm is fixed. This is the case for ML-KEM.

Note that variable-length secrets are, generally speaking, dangerous. In particular, when using key material of variable length and processing it using hash functions, a timing side channel may arise. In broad terms, when the secret is longer, the hash function may need to process more blocks internally. In some unfortunate circumstances, this has led to timing attacks, e.g., the Lucky Thirteen [LUCKY13] and Raccoon [RACCOON] attacks.

Furthermore, [AVIRAM] identifies a risk of using variable-length secrets when the hash function used in the key derivation function is no longer collision-resistant.

If concatenation were to be used with values that are not fixed-length, a length prefix or other unambiguous encoding would need to be used to ensure that the composition of the two values is injective and requires a mechanism different from that specified in this document.

Therefore, this specification **MUST** only be used with algorithms that have fixed-length shared secrets (after the variant has been fixed by the algorithm identifier in the `NamedGroup` negotiation in Section 3.1).

7. References

7.1. Normative References

[FO] Fujisaki, E. and T. Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", Journal of Cryptology, vol. 26, no. 1, pp. 80-101, DOI 10.1007/s00145-011-9114-1, December 2011, <https://doi.org/10.1007/s00145-011-9114-1>.

[HHK] Hofheinz, D., Hövelmanns, K., and E. Kiltz, "A Modular Analysis of the Fujisaki-Okamoto Transformation", Theory of Cryptography (TCC 2017), Lecture Notes in Computer Science, vol. 10677, pp. 341-371, DOI 10.1007/978-3-319-70500-2_12, 2017, <https://doi.org/10.1007/978-3-319-70500-2_12>.

[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>.

[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>.

[TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[AVIRAM] Nimrod Aviram, Benjamin Dowling, Ilan Komargodski, Kenny Paterson, Eyal Ronen, and Eylon Yogev, "[TLS] Combining Secrets in Hybrid Key Exchange in TLS 1.3", 1 September 2021, <https://mailarchive.ietf.org/arch/msg/tls/F4SVeL2xbGPaPB2GW_GkBbD_a5M/>.

[BCNS15] Bos, J., Costello, C., Naehrig, M., and D. Stebila, "Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem", 2015 IEEE Symposium on Security and Privacy, pp. 553-570, DOI 10.1109/sp.2015.40, May 2015, <https://doi.org/10.1109/sp.2015.40>.

[BERNSTEIN] Bernstein, D. J., Ed., Buchmann, J., Ed., and E. Dahmen, Ed., "Post-Quantum Cryptography", Springer Berlin, DOI 10.1007/978-3-540-88702-7, 2009, <https://doi.org/10.1007/978-3-540-88702-7>.

[BINDEL] Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., and D. Stebila, "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange", Post-Quantum Cryptography (PQCrypto 2019), Lecture Notes in Computer Science, vol. 11505, pp. 206-226, DOI 10.1007/978-3-030-25510-7_12, 2019, <https://doi.org/10.1007/978-3-030-25510-7_12>.

[CAMPAGNA] Campagna, M. and E. Crockett, "Hybrid Post-Quantum Key Encapsulation Methods (PQ KEM) for Transport Layer Security 1.2 (TLS)", Work in Progress, Internet-Draft, draft-campagna-tls-bike-sike-hybrid-07, 2 September 2021, <https://datatracker.ietf.org/doc/html/draft-campagna-tls-bike-sike-hybrid-07>.

[CECPQ1] Braithwaite, M., "Experimenting with Post-Quantum Cryptography", Google Security Blog, 7 July 2016, <https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html>.

[CECPQ2] Langley, A., "CECPQ2", 12 December 2018, <https://www.imperialviolet.org/2018/12/12/cecpq2.html>.

[DODIS] Dodis, Y. and J. Katz, "Chosen-Ciphertext Security of Multiple Encryption", Theory of Cryptography (TCC 2005), Lecture Notes in Computer Science, vol. 3378, pp. 188-209, DOI 10.1007/978-3-540-30576-7_11, 2005, <https://doi.org/10.1007/978-3-540-30576-7_11>.

[DOWLING] Dowling, B., Fischlin, M., Günther, F., and D. Stebila, "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol", Journal of Cryptology, vol. 34, article 37, DOI 10.1007/s00145-021-09384-1, July 2021, <https://doi.org/10.1007/s00145-021-09384-1>.

[ECDHE-MLKEM] Kwiatkowski, K., Kampanakis, P., Westerbaan, B., and D. Stebila, "Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3", Work in Progress, Internet-Draft, draft-kwiatkowski-tls-ecdhe-mlkem-03, 24 December 2024, <https://datatracker.ietf.org/doc/html/draft-kwiatkowski-tls-ecdhe-mlkem-03>.

[ETSI] Campagna, M., Ed., et al., "Quantum Safe Cryptography and Security: An introduction, benefits, enablers and challengers", ETSI White Paper No. 8, June 2015, <https://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf>.

[EVEN] Even, S. and O. Goldreich, "On the Power of Cascade Ciphers", Advances in Cryptology, pp. 43-50, DOI 10.1007/978-1-4684-4730-9_4, 1984, <https://doi.org/10.1007/978-1-4684-4730-9_4>.

[EXTERN-PSK] Housley, R., "TLS 1.3 Extension for Certificate-Based Authentication with an External Pre-Shared Key", RFC 8773, DOI 10.17487/RFC8773, March 2020, <https://www.rfc-editor.org/info/rfc8773>.

[FLUHRER] Fluhrer, S., "Cryptanalysis of ring-LWE based key exchange with key share reuse", Cryptology ePrint Archive, Paper 2016/085, 2016, <https://eprint.iacr.org/2016/085>.

[FRODO] Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., and D. Stebila, "Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE", Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1006-1018, DOI 10.1145/2976749.2978425, October 2016, <https://doi.org/10.1145/2976749.2978425>.

[GIACON] Giacon, F., Heuer, F., and B. Poettering, "KEM Combiners", Public-Key Cryptography (PKC 2018), Lecture Notes in Computer Science, vol. 10769, pp. 190-218, DOI 10.1007/978-3-319-76578-5_7, 2018, <https://doi.org/10.1007/978-3-319-76578-5_7>.

[HARNIK] Harnik, D., Kilian, J., Naor, M., Reingold, O., and A. Rosen, "On Robust Combiners for Oblivious Transfer and Other Primitives", Advances in Cryptology (EUROCRYPT 2005), Lecture Notes in Computer Science, vol. 3494, pp. 96-113, DOI 10.1007/11426639_6, 2005, <https://doi.org/10.1007/11426639_6>.

[HPKE] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/info/rfc9180>.

[IANA-TLS] IANA, "TLS Supported Groups", <https://www.iana.org/assignments/tls-parameters>.

[IKE-HYBRID] Tjhai, C., Tomlinson, M., Bartlett, G., Fluhrer, S., Van Geest, D., Garcia-Morchon, O., and V. Smyslov, "Framework to Integrate Post-quantum Key Exchanges into Internet Key Exchange Protocol Version 2 (IKEv2)", Work in Progress, Internet-Draft, draft-tjhai-ipsecme-hybrid-qske-ikev2-04, 9 July 2019, <https://datatracker.ietf.org/doc/html/draft-tjhai-ipsecme-hybrid-qske-ikev2-04>.

[IKE-PSK] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <https://www.rfc-editor.org/info/rfc8784>.

[KATZ] Katz, J. and Y. Lindell, "Introduction to Modern Cryptography", Third Edition, CRC Press, 2021.

[KIEFER] Kiefer, F. and K. Kwiatkowski, "Hybrid ECDHE-SIDH Key Exchange for TLS", Work in Progress, Internet-Draft, draft-kiefer-tls-ecdhe-sidh-00, 5 November 2018, <https://datatracker.ietf.org/doc/html/draft-kiefer-tls-ecdhe-sidh-00>.

[LANGLEY] Langley, A., "Post-quantum confidentiality for TLS", 11 April 2018, <https://www.imperialviolet.org/2018/04/11/pqconftls.html>.

[LUCKY13] Al Fardan, N. and K. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols", 2013 IEEE Symposium on Security and Privacy, pp. 526-540, DOI 10.1109/sp.2013.42, May 2013, <https://doi.org/10.1109/sp.2013.42>.

[NIELSEN] Nielsen, M. A. and I. L. Chuang, "Quantum Computation and Quantum Information", Cambridge University Press, 2000.

[NIST] NIST, "Post-Quantum Cryptography", <https://www.nist.gov/pqcrypto>.

[NIST-FIPS-203] NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", NIST FIPS 203, DOI 10.6028/NIST.FIPS.203, August 2024, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf>.

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

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

[OQS-102] "OQS-OpenSSL-1-0-2_stable", commit 537b2f9, 31 January 2020, <https://github.com/open-quantum-safe/openssl/tree/OQS-OpenSSL_1_0_2-stable>.

[OQS-111] "OQS-OpenSSL-1-1-1_stable", commit 5f49b7a, 8 January 2025, <https://github.com/open-quantum-safe/openssl/tree/OQS-OpenSSL_1_1_1-stable>.

[OQS-PROV] "OQS Provider for OpenSSL 3", commit 573fb25, 8 January 2026, <https://github.com/open-quantum-safe/oqs-provider/>.

[PQUIP-TERM] 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>.

[PST] Paquin, C., Stebila, D., and G. Tamvada, "Benchmarking Post-quantum Cryptography in TLS", Post-Quantum Cryptography (PQCrypto 2020), Lecture Notes in Computer Science, vol. 12100, pp. 72-91, DOI 10.1007/978-3-030-44223-1_5, 2020, <https://doi.org/10.1007/978-3-030-44223-1_5>.

[RACCOON] Merget, R., Brinkmann, M., Aviram, N., Somorovsky, J., Mittmann, J., and J. Schwenk, "Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E)", September 2020, <https://raccoon-attack.com/>.

[S2N] Hopkins, A. and M. Campagna, "Post-quantum TLS now supported in AWS KMS", AWS Security Blog, 4 November 2019, <https://aws.amazon.com/blogs/security/post-quantum-tls-now-supported-in-aws-kms/>.

[SCHANCK] Schanck, J. M. and D. Stebila, "A Transport Layer Security (TLS) Extension For Establishing An Additional Shared Secret", Work in Progress, Internet-Draft, draft-schanck-tls-additional-keyshare-00, 17 April 2017, <https://datatracker.ietf.org/doc/html/draft-schanck-tls-additional-keyshare-00>.

[WHYTE12] Schanck, J. M., Whyte, W., and Z. Zhang, "Quantum-Safe Hybrid (QSH) Ciphersuite for Transport Layer Security (TLS) version 1.2", Work in Progress, Internet-Draft, draft-whyte-qsh-tls12-02, 22 July 2016, <https://datatracker.ietf.org/doc/html/draft-whyte-qsh-tls12-02>.

[WHYTE13] Whyte, W., Zhang, Z., Fluhrer, S., and O. Garcia-Morchon, "Quantum-Safe Hybrid (QSH) Key Exchange for Transport Layer Security (TLS) version 1.3", Work in Progress, Internet-Draft, draft-whyte-qsh-tls13-06, 3 October 2017, <https://datatracker.ietf.org/doc/html/draft-whyte-qsh-tls13-06>.

[XMSS] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <https://www.rfc-editor.org/info/rfc8391>.

[ZHANG] Zhang, R., Hanaoka, G., Shikata, J., and H. Imai, "On the Security of Multiple Encryption or CCA-security+CCA-security=CCA-security?", Public Key Cryptography (PKC 2004), Lecture Notes in Computer Science, vol. 2947, pp. 360-374, DOI 10.1007/978-3-540-24632-9_26, 2004, <https://doi.org/10.1007/978-3-540-24632-9_26>.

Appendix A. Related Work

Quantum computing and post-quantum cryptography in general are outside the scope of this document. For a general introduction to quantum computing, see a standard textbook such as [NIELSEN]. For an overview of post-quantum cryptography as of 2009, see [BERNSTEIN]; while not containing more recent advances, it still provides a helpful introduction. For the current status of the NIST Post-Quantum Cryptography Standardization Project, see [NIST]. For additional perspectives on the general transition from traditional to post-quantum cryptography, see for example [ETSI], among others.

There have been several Internet-Drafts describing mechanisms for embedding post-quantum and/or hybrid key exchange in TLS:

- TLS 1.2: [WHYTE12], [CAMPAGNA]
- TLS 1.3: [KIEFER], [SCHANCK], [WHYTE13]

There have been several prototype implementations for post-quantum and/or hybrid key exchange in TLS:

- TLS 1.2: [BCNS15], [CECPQ1], [FRODO], [OQS-102], [S2N]
- TLS 1.3: [CECPQ2], [OQS-111], [OQS-PROV], [PST]

These experimental implementations have taken an ad hoc approach and not attempted to implement one of the Internet-Drafts listed above.

Unrelated to post-quantum but still related to the issue of combining multiple types of keying material in TLS is the use of pre-shared keys, especially the recent TLS Working Group document on including an external pre-shared key [EXTERN-PSK].

Considering other IETF standards, there is work on post-quantum pre-shared keys in the Internet Key Exchange Protocol Version 2 (IKEv2) [IKE-PSK] and a framework for hybrid key exchange in IKEv2 [IKE-HYBRID]. The eXtended Merkle Signature Scheme (XMSS) hash-based signature scheme has been published as an Informational RFC by the IRTF [XMSS].

In the academic literature, [EVEN] initiated the study of combining multiple symmetric encryption schemes; [ZHANG], [DODIS], and [HARNIK] examined combining multiple public key encryption schemes; and [HARNIK] coined the term "robust combiner" to refer to a compiler that constructs a hybrid scheme from individual schemes while preserving security properties. [GIACON] and [BINDEL] examined combining multiple key encapsulation mechanisms.

Acknowledgements

The ideas in this document have grown from discussions with many colleagues, including Christopher Wood, Matt Campagna, Eric Crockett, Deirdre Connolly, authors of the various hybrid documents and implementations cited in this document, and members of the TLS Working Group. The immediate impetus for this document came from discussions with attendees at the Workshop on Post-Quantum Software in Mountain View, California in January 2019. Daniel J. Bernstein and Tanja Lange commented on the risks of reuse of ephemeral public keys. Matt Campagna and the team at Amazon Web Services provided additional suggestions. Nimrod Aviram proposed restricting to fixed-length secrets.

Authors' Addresses

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

Scott Fluhrer
Cisco Systems
Email: sfluhrer@cisco.com

Shay Gueron
University of Haifa and Meta
Email: shay.gueron@gmail.com