Network Working Group   D. Stebila
Internet-Draft   University of Waterloo
Intended status: Informational   S. Fluhrer
Expires: 7 October 2024   Cisco Systems
                                                            S. Gueron
                                                             U. Haifa
                                                         5 April 2024

                       Hybrid key exchange in TLS 1.3
                      draft-ietf-tls-hybrid-design-10

Abstract

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

   Discussion of this work is encouraged to happen on the TLS IETF
   mailing list tls@ietf.org or on the GitHub repository which contains
   the draft: https://github.com/dstebila/draft-ietf-tls-hybrid-design.

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 7 October 2024.

Copyright Notice

Table of Contents

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 1.4 for more on the scope of this document.

1.1. Revision history

      RFC Editor's Note: Please remove this section prior to
      publication of a final version of this document.

Earlier versions of this document categorized various design
decisions one could make when implementing hybrid key exchange in TLS
1.3.

* draft-ietf-tls-hybrid-design-10:

   - Bump to version -10 to avoid expiry

   - Clarifications on shared secret and public key generation

* draft-ietf-tls-hybrid-design-09:

   - Remove IANA registry requests

   - Editorial changes

* draft-ietf-tls-hybrid-design-09:

   - Removal of TBD hybrid combinations using Kyber512 or secp384r1

   - Editorial changes

* draft-ietf-tls-hybrid-design-08:

   - Add reference to [SECP256R1Kyber768] and [KyberDraft00] drafts

* draft-ietf-tls-hybrid-design-07:

   - Editorial changes

   - Add reference to [X25519Kyber768] draft

* draft-ietf-tls-hybrid-design-06:

   - Bump to version -06 to avoid expiry

* draft-ietf-tls-hybrid-design-05:

   - Define four hybrid key exchange methods

   - Updates to reflect NIST's selection of Kyber

   - Clarifications and rewordings based on working group comments

* draft-ietf-tls-hybrid-design-04:

   - Some wording changes

- Remove design considerations appendix

* draft-ietf-tls-hybrid-design-03:

  - Remove specific code point examples and requested codepoint
    range for hybrid private use

  - Change "Open questions" to "Discussion"

  - Some wording changes

* draft-ietf-tls-hybrid-design-02:

  - Bump to version -02 to avoid expiry

* draft-ietf-tls-hybrid-design-01:

  - Forbid variable-length secret keys

  - Use fixed-length KEM public keys/ciphertexts

* draft-ietf-tls-hybrid-design-00:

  - Allow key_exchange values from the same algorithm to be reused
    across multiple KeyShareEntry records in the same ClientHello.

* draft-stebila-tls-hybrid-design-03:

  - Add requirement for KEMs to provide protection against key
    reuse.

  - Clarify FIPS-compliance of shared secret concatenation method.

* draft-stebila-tls-hybrid-design-02:

  - Design considerations from draft-stebila-tls-hybrid-design-00
    and draft-stebila-tls-hybrid-design-01 are moved to the
    appendix.

  - A single construction is given in the main body.

* draft-stebila-tls-hybrid-design-01:

  - Add (Comb-KDF-1) and (Comb-KDF-2) options.

  - Add two candidate instantiations.

* draft-stebila-tls-hybrid-design-00: Initial version.

1.2. 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 be able to divide
   cryptographic algorithms into two classes:

   * "Traditional" algorithms: Algorithms which are widely deployed
      today, but which may be deprecated in the future. In the context
      of TLS 1.3, examples of traditional key exchange algorithms
      include elliptic curve Diffie-Hellman using secp256r1 or x25519,
      or finite-field Diffie-Hellman.

   * "Next-generation" (or "next-gen") algorithms: Algorithms which are
      not yet widely deployed, but which may eventually be widely
      deployed. An additional facet of these algorithms may be that we
      have less confidence in their security due to them being
      relatively new or less studied. This includes "post-quantum"
      algorithms.

   "Hybrid" key exchange, in this context, means the use of two (or
   more) key exchange algorithms based on different cryptographic
   assumptions, e.g., one traditional algorithm and one next-gen
   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 of them
   is post-quantum, this is a Post-Quantum Traditional Hybrid Scheme
   [I-D.driscoll-pqt-hybrid-terminology]; while this is the initial use
   case for this draft, we do not limit this draft to that case. We use
   the term "component" algorithms to refer to the algorithms combined
   in a hybrid key exchange.

   We note that some authors prefer the phrase "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 composite 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 we opt for
the more generic term "next-generation" algorithms rather than
exclusively "post-quantum" algorithms.

Note that TLS 1.3 uses the phrase "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 not Diffie-Hellman-based nor a group.

1.3. 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 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, and 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: 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.

1.4. 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. This selection will be based on the recommendations
   by the Crypto Forum Research Group (CFRG), which is currently
   waiting for the results of the NIST Post-Quantum Cryptography
   Standardization Project [NIST].

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

1.5. Goals

The primary goal of a hybrid key exchange mechanism is to facilitate
the establishment of a shared secret which remains secure as long as
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
   middle-boxes 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 traditional shared secret (assuming the
       hybrid-aware client is willing to downgrade to traditional-
       only).

   3. Non-hybrid-aware client, hybrid-aware server: These parties
       should establish a traditional shared secret (assuming the
       hybrid-aware server is willing to downgrade to traditional-
       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.

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 / secret key pair can be treated as a long-term key or
reused. 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, 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 e.g.
[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. (We note that this is independent from future documents standardizing solely post-quantum key exchange methods, which would have to be assigned their own identifier.)

Specific values shall be registered by IANA in the TLS Supported Groups registry.

```
        enum {

                /* Elliptic Curve Groups (ECDHE) */
                secp256r1(0x0017), secp384r1(0x0018), secp521r1(0x0019),
                x25519(0x001D), x448(0x001E),

                /* Finite Field Groups (DHE) */
                ffdhe2048(0x0100), ffdhe3072(0x0101), ffdhe4096(0x0102),
                ffdhe6144(0x0103), ffdhe8192(0x0104),

                /* Hybrid Key Exchange Methods */
                ...,

                /* Reserved Code Points */
                ffdhe_private_use(0x01FC..0x01FF),
                ecdhe_private_use(0xFE00..0xFEFF),
                (0xFFFF)
        } NamedGroup;
```

3.2. Transmitting public keys and ciphertexts

We take 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 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.

[TLS13] requires that ``The key_exchange values for each
KeyShareEntry MUST be generated independently.'' In the context of
this document, since the same algorithm may appear in multiple named
groups, we relax the above requirement to allow the same key_exchange
value for the same algorithm to be reused in multiple KeyShareEntry
records sent in 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 we also take 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, we do not add any
   additional structure (length fields) in the concatenation procedure:
   for both the traditional groups and Kyber, 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. [NIST-SP-800-56C] or [NIST-SP-800-135] give NIST 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 secret is from an approved method.

4. Discussion

   Larger public keys and/or ciphertexts. The HybridKeyExchange struct
   in Section 3.2 limits public keys and ciphertexts to 2^16-1 bytes;
   this is bounded by the same (2^16-1)-byte limit on the key_exchange
   field in the KeyShareEntry struct. Some post-quantum KEMs have
   larger public keys and/or ciphertexts; for example, Classic
   McEliece's smallest parameter set has public key size 261,120 bytes.
   However, all defined parameter sets for Kyber have public keys and
   ciphertexts that fall within the TLS constraints.

   Duplication of key shares. Concatenation of public keys in the
   HybridKeyExchange struct as described in Section 3.2 can result in
   sending duplicate key shares. For example, if a client wanted to
   offer support for two combinations, say "SecP256r1Kyber768Draft00"
   and "X25519Kyber768Draft00", it would end up sending two kyber768
   public keys, since the KeyShareEntry for each combination contains
   its own copy of a kyber768 key. This duplication may be more
   problematic for post-quantum algorithms which have larger public
   keys. On the other hand, if the client wants to offer, for example
   "SecP256r1Kyber768Draft00" and "secp256r1" (for backwards
   compatibility), there is relatively little duplicated data (as the
   secp256r1 keys are comparatively small).

   Failures. Some post-quantum key exchange algorithms, including
   Kyber, have non-zero probability of failure, meaning two honest
   parties may derive different shared secrets. This would cause a
   handshake failure. Kyber has a cryptographically small failure rate;
   if other algorithms are used, implementers should be aware of the
   potential of handshake failure. Clients can retry if a failure is
   encountered.

5. IANA Considerations

   IANA will assign identifiers from the TLS Supported Groups section
   for the hybrid combinations defined following this document. These
   assignments should be made in a range that is distinct from the
   Elliptic Curve Groups and the Finite Field Groups ranges.

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 from 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.
Kyber has such security properties. However, some other post-quantum
KEMs are 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 Kyber.

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] identified 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 which
have fixed-length shared secrets (after the variant has been fixed by
the algorithm identifier in the NamedGroup negotiation in
Section 3.1).

7. Acknowledgements

8. References

8.1. Normative References

   [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>.

   [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/rfc/rfc8446>.

8.2. 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, DOI 10.1109/sp.2015.40, May 2015,
               <https://doi.org/10.1109/sp.2015.40>.

   [BERNSTEIN]
               "Post-Quantum Cryptography", Springer Berlin
               Heidelberg book, 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 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", 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 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, no. 4,
               DOI 10.1007/s00145-021-09384-1, July 2021,
               <https://doi.org/10.1007/s00145-021-09384-1>.

   [ETSI]   Campagna, M., Ed. and others, "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/rfc/rfc8773>.

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

   [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>.

   [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,
              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 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", Lecture Notes in Computer Science pp.
              96-113, DOI 10.1007/11426639_6, 2005,
              <https://doi.org/10.1007/11426639_6>.

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

   [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/rfc/rfc9180>.

   [I-D.driscoll-pqt-hybrid-terminology]
              D, F., "Terminology for Post-Quantum Traditional Hybrid
              Schemes", Work in Progress, Internet-Draft, draft-
              driscoll-pqt-hybrid-terminology-02, 7 March 2023,
              <https://datatracker.ietf.org/doc/html/draft-driscoll-pqt-
              hybrid-terminology-02>.

   [IKE-HYBRID]
             Tjhai, C., Tomlinson, M., grbartle@cisco.com, 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/rfc/rfc8784>.

   [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>.

   [KyberDraft00]
             Schwabe, P. and B. Westerbaan, "Kyber Post-Quantum KEM",
             Work in Progress, Internet-Draft, draft-cfrg-schwabe-
             kyber-04, 2 January 2024,
             <https://datatracker.ietf.org/doc/html/draft-cfrg-schwabe-
             kyber-04>.

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

   [LUCKY13] Al Fardan, N. J. and K. G. Paterson, "Lucky Thirteen:
             Breaking the TLS and DTLS record protocols", n.d.,
             <https://ieeexplore.ieee.org/
             iel7/6547086/6547088/06547131.pdf>.

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

   [NIST]   National Institute of Standards and Technology (NIST),
             "Post-Quantum Cryptography", n.d.,
             <https://www.nist.gov/pqcrypto>.

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

   [NIST-SP-800-56C]
             National Institute of Standards and Technology (NIST),
             "Recommendation for Key-Derivation Methods in Key-
             Establishment Schemes", August 2020,
             <https://doi.org/10.6028/NIST.SP.800-56Cr2>.

   [OQS-102] Open Quantum Safe Project, "OQS-OpenSSL-1-0-2_stable",
             November 2018, <https://github.com/open-quantum-
             safe/openssl/tree/OQS-OpenSSL_1_0_2-stable>.

   [OQS-111] Open Quantum Safe Project, "OQS-OpenSSL-1-1-1_stable",
             January 2022, <https://github.com/open-quantum-
             safe/openssl/tree/OQS-OpenSSL_1_1_1-stable>.

   [OQS-PROV] Open Quantum Safe Project, "OQS Provider for OpenSSL 3",
             July 2023,
             <https://github.com/open-quantum-safe/oqs-provider/>.

   [PST]   Paquin, C., Stebila, D., and G. Tamvada, "Benchmarking
             Post-quantum Cryptography in TLS", Post-Quantum
             Cryptography 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]   Amazon Web Services, "Post-quantum TLS now supported in
             AWS KMS", 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>.

   [SECP256R1Kyber768]
             Kwiatkowski, K. and P. Kampanakis, "Post-quantum hybrid
             ECDHE-Kyber Key Agreement for TLSv1.3", Work in Progress,
             Internet-Draft, draft-kwiatkowski-tls-ecdhe-kyber-01, 18
             May 2023, <https://datatracker.ietf.org/doc/html/draft-
             kwiatkowski-tls-ecdhe-kyber-01>.

   [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>.

   [X25519Kyber768]
             Westerbaan, B. and D. Stebila, "X25519Kyber768Draft00
             hybrid post-quantum key agreement", Work in Progress,
             Internet-Draft, draft-tls-westerbaan-xyber768d00-03, 24
             September 2023, <https://datatracker.ietf.org/doc/html/
             draft-tls-westerbaan-xyber768d00-03>.

   [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/rfc/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 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].
   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:

*   Internet-Drafts for TLS 1.2: [WHYTE12], [CAMPAGNA]

*   Internet-Drafts for TLS 1.3: [KIEFER], [SCHANCK], [WHYTE13]

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

*   Experimental implementations in TLS 1.2: [BCNS15], [CECPQ1], [FRODO], [OQS-102], [S2N]

*   Experimental implementations in 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 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 preshared keys in IKEv2 [IKE-PSK] and a framework for hybrid key exchange in IKEv2 [IKE-HYBRID]. The 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.

Authors' Addresses

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

   Scott Fluhrer
   Cisco Systems

      Email: sfluhrer@cisco.com

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