Hybrid key exchange in TLS 1.3

draft-stebila-tls-hybrid-design

Douglas Stebila, Scott Fluhrer, Shay Gueron

https://dstebila.github.io/draft-stebila-tls-hybrid-design/

UNIVERSITY OF WATERLOO

NSERC CRSNG

IETF 105 TLSWG • 2019-07-25

Motivation and Goals

- Multiple sources of interest in using multiple key exchange algorithms simultaneously as part of transition to post-quantum crypto
  - Several Internet-Drafts already:
    - TLS 1.2: Schanck, Whyte, Zhang 2016; Amazon 2019
    - TLS 1.3: Shack, Stebila 2017; Whyte, Zhang, Fluhrer, Garcia-Morchon 2017; Kiefer, Kwiatkowski 2018
  - Experimental implementations: Google CECPQ1, CECPQ2; Open Quantum Safe; …
- Need PQ key exchange before we need PQ authentication because future quantum computers could retroactively decrypt, but not retroactively impersonate
- **Goal: develop framework in which key exchange in TLS 1.3 can be extended with additional keyshares**
  - **Should this be Informational? Experimental? Proposed standard?**

Non-Goals

- Selecting or specifying one or more post-quantum algorithms to actually use in TLS

Draft-00 @ IETF 104

Contained a “menu” of design options along several axes

1. How to negotiate which algorithms?
2. How many algorithms?
3. How to transmit public key shares?
4. How to combine secrets?

Feedback from working group:

- Avoid changes to key schedule
- Present one or two instantiations
- Specific feedback on some aspects

Draft-01 @ IETF 105

Kept menu of design choices

Constructed two candidate instantiations from menu for discussion

1. Directly negotiate each hybrid algorithm; separate key shares
2. Code points for pre-defined combinations; concatenated key shares

Additional KDF-based options for combining keys

Candidate Instantiation 1 – Negotiation

Follows draft-whyte-qsh-tls13-06

NamedGroup enum for supported_groups extension contains "hybrid markers" with no pre-defined meaning

Each hybrid marker points to a mapping in an extension, which lists which combinations the client proposes; between 2 and 10 algorithms permitted

supported_groups:
hybrid_marker00, hybrid_marker01, hybrid_marker02, secp256r1

HybridExtension:

- hybrid_marker00 → secp256r1+sike123+ntru456
- hybrid_marker01 → secp256r1+sike123
- hybrid_marker02 → secp256r1+ntru456

Candidate Instantiation 1 – Conveying keyshares

Client's key shares:

- Existing KeyShareClientHello allows multiple key shares
- => Send 1 key share per algorithm
  - secp256r1, sike123, ntru456
- No changes required to data structures or logic

Server's key shares:

- Respond with NamedGroup = hybrid_markerXX
- Existing KeyShareServerHello only permits one key share
- => Squeeze 2+ key shares into single key share field by concatenation

```
struct {
    KeyShareEntry key_share<2..10>;
} HybridKeyShare;
```

Candidate Instantiation 1 – Combining keys

```
                                  0
                                  |
                                  v
                    PSK ->  HKDF-Extract = Early Secret
                                  |
                                  +-----> Derive-Secret(...)
                                  +-----> Derive-Secret(...)
                                  +-----> Derive-Secret(...)
                                  |
                                 v
concatenated      0
shared            |
secret   -> HKDF-Extract        Derive-Secret(., "derived", "")
^^^^^^            |               |
                  v               v
              output  ------> 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(...)
```

Candidate Instantiation 2 – Negotiation

Follows draft-kiefer-tls-ecdhe-sidh-00, Open Quantum Safe implementation, ...

New NamedGroup element standardized for each desired combination

No internal structure to new code points

```
enum {
      /* existing named groups */
      secp256r1 (23),
      x25519 (0x001D),
      ...,

      /* new code points eventually defined for post-quantum algorithms */
      PQ1 (0x????),
      PQ2 (0x????),
      ...,

      /* new code points defined for hybrid combinations */
      secp256r1_PQ1 (0x????),
      secp256r1_PQ2 (0x????),
      x25519_PQ1 (0x????),
      x25519_PQ2 (0x????),

      /* existing reserved code points */
      ffdhe_private_use (0x01FC..0x01FF),
      ecdhe_private_use (0xFE00..0xFEFF),
      (0xFFFF)
 } NamedGroup;
```

Candidate Instantiation 2 – Conveying keyshares

KeyShareClientHello contains an entry for each code point listed in supported_groups

KeyShareServerHello contains a single entry for the chosen code point

KeyShareEntry for hybrid code points is an opaque string parsed with the following internal structure:

```
struct {
    KeyShareEntry key_share<2..10>;
} HybridKeyShare;
```

Candidate Instantiation 1

Adds new negotiation logic and ClientHello extensions

Does not result in duplicate key shares or combinatorial explosion of NamedGroups

Candidate Instantiation 2

No change in negotiation logic or data structures

No change to protocol logic: concatenation of key shares and KDFing shared secrets can be handled “internally” to a method

Results in combinatorial explosion of NamedGroups

Duplicate key shares will be sent

Next steps?

1. Produce an Informational document that outlines different options and possible instantiations

   - or -

2. Produce an Experimental / Proposed Standard describing a single instantiation
    a. How to decide among current options? Experiments? Further discussion?

Hybrid key exchange in TLS 1.3

draft-stebila-tls-hybrid-design

Douglas Stebila, Scott Fluhrer, Shay Gueron

https://dstebila.github.io/draft-stebila-tls-hybrid-design/

UNIVERSITY OF WATERLOO

NSERC CRSNG

IETF 105 TLSWG • 2019-07-25