Network Working Group
Internet-Draft
Intended status: Experimental
Expires: October 19, 2017

J. Schanck
University of Waterloo
D. Stebila
McMaster University

April 17, 2017

A Transport Layer Security (TLS) Extension For Establishing An Additional Shared Secret
draft-schanck-tls-additional-keyshare-00

Abstract

This document defines a Transport Layer Security (TLS) extension that allows parties to establish an additional shared secret using a second key exchange algorithm and incorporates this shared secret into the TLS key schedule.

Status of This Memo

Copyright Notice

Table of Contents

```
   1.  Introduction  . . . . . . . . . . . . . . . . . . . . . . . .   2
     1.1.  Use cases . . . . . . . . . . . . . . . . . . . . . . . .   3
     1.2.  Conventions and Terminology . . . . . . . . . . . . . . .   3
   2.  Additional Key Share Extension  . . . . . . . . . . . . . . .   4
     2.1.  ExtensionType . . . . . . . . . . . . . . . . . . . . . .   4
     2.2.  Existing data structures  . . . . . . . . . . . . . . . .   4
     2.3.  AdditionalKeyShare extension  . . . . . . . . . . . . . .   4
       2.3.1.  Requirements for use in ClientHello . . . . . . . . .   5
       2.3.2.  Requirements for use in ServerHello . . . . . . . . .   6
       2.3.3.  Requirements for use in HelloRetryRequest . . . . . .   6
     2.4.  Backward Compatibility  . . . . . . . . . . . . . . . . .   6
       2.4.1.  Negotiating with a server that does not support the
               additional_key_share extension  . . . . . . . . . . .   6
       2.4.2.  Negotiating with a client that does not support the
               additional_key_share extension  . . . . . . . . . . .   6
   3.  Key Schedule  . . . . . . . . . . . . . . . . . . . . . . . .   7
   4.  Pre-shared key modes and session resumption . . . . . . . . .   8
   5.  Security Considerations . . . . . . . . . . . . . . . . . . .   9
   6.  IANA Considerations . . . . . . . . . . . . . . . . . . . . .   9
   7.  References  . . . . . . . . . . . . . . . . . . . . . . . . .   9
     7.1.  Normative References  . . . . . . . . . . . . . . . . . .   9
     7.2.  Informative References  . . . . . . . . . . . . . . . . .  10
   Authors' Addresses  . . . . . . . . . . . . . . . . . . . . . . .  10
```

1. Introduction

The key schedule of TLS 1.3 [TLS13draft19] is capable of extracting session key material from multiple shared secrets. A notable use of this feature is in forward secret pre-shared key (PSK) modes wherein a PSK is combined with an ephemeral shared secret established through a Diffie-Hellman exchange. While the key schedule can process arbitrary lists of shared secrets, it is currently only possible to incorporate a pre-shared key and a shared secret established through the "key_share" extension.

This document defines a TLS ClientHello, HelloRetryRequest, and ServerHello extension of type "additional_key_share" that allows parties to establish an additional shared secret.

This document does not define any named groups or key exchange modes.

1.1. Use cases

One use case is to provide pre- to post-quantum transitional security while hedging against potential weaknesses of post-quantum algorithms. A post-quantum cryptographic algorithm is one that is believed to be resistant to attacks by quantum computers; algorithms based on factoring and discrete log are said to be pre-quantum. Authenticated and confidential channel establishment protocols are said to be secure in a transitional setting if they provide pre-quantum authentication and post-quantum confidentiality. Such protocols provide forward secrecy so long as adversaries do not have quantum computers at the time of session establishment.

An additional key share can be used to combine a high-confidence pre-quantum confidentiality mechanism with a more experimental post-quantum confidentiality mechanism without any added risk.

One could argue that if post-quantum algorithms are available then they should be used in place of pre-quantum algorithms. However there are several reasons why a user might not want to rely solely on a post-quantum algorithm today. First, confidence in cryptographic assumptions relies in part on the duration and intensity of their study. Most post-quantum assumptions have received less scrutiny than DH or ECDH, and cryptanalysis may progress rapidly as more attention is drawn to these assumptions. Second, the cryptographic community has less experience writing secure implementations of post-quantum algorithms, and one may be concerned that there are yet-to-be-discovered implementation pitfalls and side-channel attacks that could compromise confidentiality. Finally there may be users who are required to use certain pre-quantum algorithms, but who nevertheless desire forward secrecy against post-quantum adversaries. For example, NIST has made it clear that hybrid modes are not incompatible with FIPS 140 validation [NISTPQFAQ].

The simultaneous use of pre-quantum and post-quantum algorithms provides users with the potential of long-term, quantum-resistant confidentiality without any added risk.

1.2. Conventions and 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 RFC 2119 [RFC2119].

In [TLS13draft19], all asymmetric key exchange modes are either (finite field) Diffie-Hellman or elliptic curve Diffie-Hellman, and the term "named group" is used to identify which group. Some key

exchange mechanisms, especially post-quantum mechanisms, are not parameterized by a group. However, we continue to use the term "named group" to identify a key exchange mechanism more broadly.

2. Additional Key Share Extension

The "additional_key_share" extension replicates the functionality of the "key_share" extension defined in [TLS13draft19], although there are some semantic differences between the two extensions.

2.1. ExtensionType

This document extends the ExtensionType enum as follows:

```
enum {
    ...,
    additional_key_share(XX),
    (65535)
} ExtensionType;
```

2.2. Existing data structures

The definition of the AdditionalKeyShare extension requires the KeyShareEntry and NamedGroup structs defined in [TLS13draft19]. NamedGroup is a 2-octet enum. For convenience of the reader, we reproduce the definition of KeyShareEntry from [TLS13draft19] below:

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

2.3. AdditionalKeyShare extension

The "extension_data" field of the "additional_key_share" extension contains an "AdditionalKeyShare" value.

```
      struct {
        select (Handshake.msg_type) {
            case client_hello:
                KeyShareEntry additional_client_shares<0..2^16-1>;

            case hello_retry_request:
                NamedGroup additional_selected_group;

            case server_hello:
                KeyShareEntry additional_server_share;
        };
      } AdditionalKeyShare;
```

additional_client_shares A list of offered KeyShareEntry values in descending order of client preference.

additional_selected_group A mutually supported key exchange algorithm that the server is willing to negotiate if the client sends an "additional_key_share" extension in a new ClientHello.

additional_server_share A single KeyShareEntry value that is in the same NamedGroup as one of the client's shares.

2.3.1. Requirements for use in ClientHello

The client MUST NOT send the "additional_key_share" extension without sending the "key_share" extension. The client MAY send an empty "additional_client_shares" vector when requesting a HelloRetryRequest, but SHOULD NOT do so otherwise.

The client MUST NOT offer a KeyShareEntry value for a NamedGroup that is not listed in the "supported_groups" extension.

The client MUST NOT offer multiple KeyShareEntry values in the "additional_client_shares" vector for the same NamedGroup. This is because the "additional_server_share" value of the server's "additional_key_share" extension must uniquely identify the client share to which it corresponds.

The client MAY offer a KeyShareEntry value in "additional_client_shares" for a NamedGroup that they offered in their "key_share" extension, as this causes no ambiguity.

Each value in the client's "additional_client_shares" vector MUST be generated independently. The independence requirement extends to the KeyShareEntry values offered in the "key_share" extension. In particular, the client MUST NOT offer KeyShareEntry values in

"additional_key_share" that duplicate the contents of KeyShareEntry values offered in "key_share".

The server MAY check for violations of these rules and MAY abort the connection with a fatal "illegal_parameter" alert if a rule is violated.

2.3.2. Requirements for use in ServerHello

The server MUST NOT send a KeyShareEntry for any NamedGroup not indicated in the "supported_groups" extension. The server MUST NOT send a KeyShareEntry for a NamedGroup that does not match one of the client's offers.

2.3.3. Requirements for use in HelloRetryRequest

The server MAY send HelloRetryRequest based on the contents of the client's "additional_client_shares" vector. The client processes this message as in Section 4.2.5 of [TLS13draft19].

2.4. Backward Compatibility

2.4.1. Negotiating with a server that does not support the additional_key_share extension

If a server does not provide an "additional_key_share" extension in its ServerHello, the client MAY abort the handshake with a "missing_extension" alert, or the client MAY finish the handshake based on the server's "key_share" extension. (The latter allows a client to negotiate a connection with a server who does not recognize this extension without retrying the handshake.)

2.4.2. Negotiating with a client that does not support the additional_key_share extension

If a client does not provide an "additional_key_share" extension in its ClientHello, the server MAY abort the handshake with a "missing_extension" alert, or the server MAY finish the handshake based on the client's "key_share" extension.

If the server chooses to finish the handshake based on the client's "key_share" extension, this allows a server that supports both pre-quantum and post-quantum key exchange to negotiate a connection with a client that does not recognize this extension and only supports pre-quantum key exchange.

3. Key Schedule

The presence of the "additional_key_share" extension alters the derivation of the master secret. The key schedule employed by TLS 1.3 handles a list of input secrets by iteratively invoking HKDF-Extract. When the "additional_key_share" extension is not present, secrets are processed in the following order:

- PSK

- (EC)DHE shared secret.

When an additional secret is derived through "additional_key_share" the order is:

- PSK

- (EC)DHE shared secret

- Additional secret.

```
                 0
                 |
                 v
     PSK ->  HKDF-Extract = Early Secret
                 |
                 +--> Derive-Secret(...) = binder_key
                 +--> Derive-Secret(...) = client_early_traffic_secret
                 +--> Derive-Secret(...) = early_exporter_secret
                 |
                 v
               Derive-Secret(., "derived secret", "")
                 |
                 v
  (EC)DHE -> HKDF-Extract
                 |
                 v
               Derive-Secret(., "derived secret", "")
                 |
                 |
 Additional      v
   Secret -> HKDF-Extract = Handshake Secret
                 |
                 +--> Derive-Secret(...) = client_handshake_traffic_secret
                 +--> Derive-Secret(...) = server_handshake_traffic_secret
                 |
                 v
               Derive-Secret(., "derived secret", "")
                 |
                 v
        0 -> HKDF-Extract = Master Secret
                 |
                 +--> Derive-Secret(...) = client_traffic_secret_0
                 +--> Derive-Secret(...) = server_traffic_secret_0
                 +--> Derive-Secret(...) = exporter_secret
                 +--> Derive-Secret(...) = resumption_master_secret
```

4. Pre-shared key modes and session resumption

[[FOR DISCUSSION]]

TLS 1.3 allows the client to restrict the use of PSKs that they provide in ClientHello through the "psk_key_exchange_modes" extension. The client may, for instance, request that the PSK only be used in a PSK+(EC)DHE mode, so as to ensure that the resumed session has forward secrecy.

If the client sends "additional_key_share" in an initial ClientHello, it is reasonable to expect that they will want to use

"additional_key_share" in PSK-resumption. It is possible to accomodate such a client by defining a new PskKeyExchangeMode, however there is a caveat in doing so that we feel it is worth pointing out.

Suppose that a PSK has been established through some combination of pre-quantum and post-quantum mechanisms, as in our proposed use case. This PSK is treated as long-term key material during resumption, so a "psk_dhe_ke" mode would not be sufficient to preserve the security properties of the initial handshake, namely forward secrecy against post-quantum adversaries. To avoid this, a post-quantum mechanisms must be used in the resumption handshake.

It is not sufficient to require the use of "additional_key_share" during resumption, as this could be used to combine two pre-quantum mechanisms.

Possible remedies:

- o Add a new PskKeyExchangeMode that enforces the use of the same NamedGroups that were used to establish the initial secret. enum { ..., psk_same_groups_ke(XX), (255) } PskKeyExchangeMode;

- o Add a new PskKeyExchangeMode for "transitional" security enum { ..., psk_transitional_ke(XX), (255) } PskKeyExchangeMode;

5. Security Considerations

This document does not change the intended security properties of TLS. In particular, it retains the goals of "establishing the same session key" and "secrecy of the session key" as described in Appendix E.1 of [TLS13draft19].

6. IANA Considerations

IANA [SHALL add/has added] a new entry to the TLS extensions ExtensionType values registry for "additional_key_share".

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[NISTPQFAQ]
NIST, "Post-Quantum Crypto Standardization FAQ", April 2017, <http://csrc.nist.gov/groups/ST/post-quantum-crypto/faq.html#Q1>.

[TLS13draft19]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", March 2017, <https://tools.ietf.org/html/draft-ietf-tls-tls13-19>.

Authors' Addresses

John M. Schanck
University of Waterloo
200 University Avenue West
Waterloo, ON N2L 3G1
Canada

Email: jschanck@uwaterloo.ca

Douglas Stebila
McMaster University
1280 Main Street West
Hamilton, ON L8S 4L8
Canada

Email: stebilad@mcmaster.ca