        TLS 1.3 Extension for Certificate-based Authentication with an External
                              Pre-Shared Key
                 draft-ietf-tls-tls13-cert-with-extern-psk-07

Abstract

   This document specifies a TLS 1.3 extension that allows a server to
   authenticate with a combination of a certificate and an external pre-
   shared key (PSK).

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 June 25, 2020.

1. Introduction

The TLS 1.3 [RFC8446] handshake protocol provides two mutually
exclusive forms of server authentication. First, the server can be
authenticated by providing a signature certificate and creating a
valid digital signature to demonstrate that it possesses the
corresponding private key. Second, the server can be authenticated
by demonstrating that it possesses a pre-shared key (PSK) that was
established by a previous handshake. A PSK that is established in
this fashion is called a resumption PSK. A PSK that is established
by any other means is called an external PSK. This document
specifies a TLS 1.3 extension permitting certificate-based server
authentication to be combined with an external PSK as an input to the
TLS 1.3 key schedule.

Several implementors wanted to gain more experience with this
specification before producing a standards-track RFC. As a result,
this specification is being published as an Experimental RFC to
enable interoperable implementations and gain deployment and
operational experience.

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.

3. Motivation and Design Rationale

The development of a large-scale quantum computer would pose a
serious challenge for the cryptographic algorithms that are widely
deployed today, including the digital signature algorithms that are
used to authenticate the server in the TLS 1.3 handshake protocol.
It is an open question whether or not it is feasible to build a
large-scale quantum computer, and if so, when that might happen.
However, if such a quantum computer is invented, many of the
cryptographic algorithms and the security protocols that use them
would become vulnerable.

The TLS 1.3 handshake protocol employs key agreement algorithms and
digital signature algorithms that could be broken by the development
of a large-scale quantum computer [I-D.hoffman-c2pq]. The key
agreement algorithms include Diffie-Hellman (DH) [DH1977] and
Elliptic Curve Diffie-Hellman (ECDH) [IEEE1363]; the digital
signature algorithms include RSA [RFC8017] and Elliptic Curve Digital
Signature Algorithm (ECDSA) [FIPS186]. As a result, an adversary

that stores a TLS 1.3 handshake protocol exchange today could decrypt
the associated encrypted communications in the future when a large-
scale quantum computer becomes available.

In the near-term, this document describes TLS 1.3 extension to
protect today's communications from the future invention of a large-
scale quantum computer by providing a strong external PSK as an input
to the TLS 1.3 key schedule while preserving the authentication
provided by the existing certificate and digital signature
mechanisms.

4. Extension Overview

This section provides a brief overview of the
"tls_cert_with_extern_psk" extension.

The client includes the "tls_cert_with_extern_psk" extension in the
ClientHello message. The "tls_cert_with_extern_psk" extension MUST
be accompanied by the "key_share", "psk_key_exchange_modes", and
"pre_shared_key" extensions. The client MAY also find it useful to
include the "supported_groups" extension. Since the
"tls_cert_with_extern_psk" extension is intended to be used only with
initial handshakes, it MUST NOT be sent alongside the "early_data"
extension. These extensions are all described in Section 4.2 of
[RFC8446], which also requires the "pre_shared_key" extension to be
the last extension in the ClientHello message.

If the client includes both the "tls_cert_with_extern_psk" extension
and the "early_data" extension, then the server MUST terminate the
connection with an "illegal_parameter" alert.

If the server is willing to use one of the external PSKs listed in
the "pre_shared_key" extension and perform certificate-based
authentication, then the server includes the
"tls_cert_with_extern_psk" extension in the ServerHello message. The
"tls_cert_with_extern_psk" extension MUST be accompanied by the
"key_share" and "pre_shared_key" extensions. If none of the external
PSKs in the list provided by the client is acceptable to the server,
then the "tls_cert_with_extern_psk" extension is omitted from the
ServerHello message.

When the "tls_cert_with_extern_psk" extension is successfully
negotiated, the TLS 1.3 key schedule processing includes both the
selected external PSK and the (EC)DHE shared secret value. As a
result, the Early Secret, Handshake Secret, and Master Secret values
all depend upon the value of the selected external PSK. Of course,
the Early Secret does not depend upon the (EC)DHE shared secret.

The authentication of the server and optional authentication of the
client depend upon the ability to generate a signature that can be
validated with the public key in their certificates. The
authentication processing is not changed in any way by the selected
external PSK.

Each external PSK is associated with a single hash algorithm, which
is required by Section 4.2.11 of [RFC8446]. The hash algorithm MUST
be set when the PSK is established, with a default of SHA-256.

5. **Certificate with External PSK Extension**

This section specifies the "tls_cert_with_extern_psk" extension,
which MAY appear in the ClientHello message and ServerHello message.
It MUST NOT appear in any other messages. The
"tls_cert_with_extern_psk" extension MUST NOT appear in the
ServerHello message unless the "tls_cert_with_extern_psk" extension
appeared in the preceding ClientHello message. If an implementation
recognizes the "tls_cert_with_extern_psk" extension and receives it
in any other message, then the implementation MUST abort the
handshake with an "illegal_parameter" alert.

The general extension mechanisms enable clients and servers to
negotiate the use of specific extensions. Clients request extended
functionality from servers with the extensions field in the
ClientHello message. If the server responds with a HelloRetryRequest
message, then the client sends another ClientHello message as
described in Section 4.1.2 of [RFC8446], including the same
"tls_cert_with_extern_psk" extension as the original ClientHello
message, or aborts the handshake.

Many server extensions are carried in the EncryptedExtensions
message; however, the "tls_cert_with_extern_psk" extension is carried
in the ServerHello message. Successful negotiation of the
"tls_cert_with_extern_psk" extension affects the key used for
encryption, so it cannot be carried in the EncryptedExtensions
message. Therefore, the "tls_cert_with_extern_psk" extension is only
present in the ServerHello message if the server recognizes the
"tls_cert_with_extern_psk" extension and the server possesses one of
the external PSKs offered by the client in the "pre_shared_key"
extension in the ClientHello message.

The Extension structure is defined in [RFC8446]; it is repeated here
for convenience.

```
   struct {
       ExtensionType extension_type;
       opaque extension_data<0..2^16-1>;
   } Extension;
```

The "extension_type" identifies the particular extension type, and
the "extension_data" contains information specific to the particular
extension type.

This document specifies the "tls_cert_with_extern_psk" extension,
adding one new type to ExtensionType:

```
   enum {
       tls_cert_with_extern_psk(TBD), (65535)
   } ExtensionType;
```

The "tls_cert_with_extern_psk" extension is relevant when the client
and server possess an external PSK in common that can be used as an
input to the TLS 1.3 key schedule. The "tls_cert_with_extern_psk"
extension is essentially a flag to use the external PSK in the key
schedule, and it has the following syntax:

```
   struct {
       select (Handshake.msg_type) {
           case client_hello: Empty;
           case server_hello: Empty;
       };
   } CertWithExternPSK;
```

5.1. Companion Extensions

Section 4 lists the extensions that are required to accompany the
"tls_cert_with_extern_psk" extension. Most of those extensions are
are not impacted in any way by this specification. However, this
section discusses the extensions that require additional
consideration.

The "psk_key_exchange_modes" extension is defined in Section 4.2.9 of
[RFC8446]. The "psk_key_exchange_modes" extension restricts the use
of both the PSKs offered in this ClientHello and those that the
server might supply via a subsequent NewSessionTicket. As a result,
when the "psk_key_exchange_modes" extension is included in the
ClientHello message, clients MUST include psk_dhe_ke mode. In

addition, clients MAY also include psk_ke mode to support a
subsequent NewSessionTicket. When the "psk_key_exchange_modes"
extension is included in the ServerHello message, servers MUST select
the psk_dhe_ke mode for the initial handshake. Servers MUST select a
key exchange mode that is listed by the client for subsequent
handshakes that include the resumption PSK from the initial
handshake.

The "pre_shared_key" extension is defined in Section 4.2.11 of
[RFC8446]. The syntax is repeated below for convenience. All of the
listed PSKs MUST be external PSKs. If a resumption PSK is listed
along with the "tls_cert_with_extern_psk" extension, the server MUST
abort the handshake with an "illegal_parameter" alert.

```
  struct {
      opaque identity<1..2^16-1>;
      uint32 obfuscated_ticket_age;
  } PskIdentity;

  opaque PskBinderEntry<32..255>;

  struct {
      PskIdentity identities<7..2^16-1>;
      PskBinderEntry binders<33..2^16-1>;
  } OfferedPsks;

  struct {
      select (Handshake.msg_type) {
          case client_hello: OfferedPsks;
          case server_hello: uint16 selected_identity;
      };
  } PreSharedKeyExtension;
```

The OfferedPsks contains the list of PSK identities and associated
binders for the external PSKs that the client is willing to use with
the server.

The identities are a list of external PSK identities that the client
is willing to negotiate with the server. Each external PSK has an
associated identity that is known to the client and the server; the
associated identities may be known to other parties as well. In
addition, the binder validation (see below) confirms that the client
and server have the same key associated with the identity.

The obfuscated_ticket_age is not used for external PSKs. As stated in Section 4.2.11 of [RFC8446], clients SHOULD set this value to 0, and servers MUST ignore the value.

The binders are a series of HMAC [RFC2104] values, one for each external PSK offered by the client, in the same order as the identities list. The HMAC value is computed using the binder_key, which is derived from the external PSK, and a partial transcript of the current handshake. Generation of the binder_key from the external PSK is described in Section 7.1 of [RFC8446]. The partial transcript of the current handshake includes a partial ClientHello up to and including the PreSharedKeyExtension.identities field as described in Section 4.2.11.2 of [RFC8446].

The selected_identity contains the index of the external PSK identity that the server selected from the list offered by the client. As described in Section 4.2.11.2 of [RFC8446], the server MUST validate the binder value that corresponds to the selected external PSK, and if the binder does not validate, the server MUST abort the handshake with an "illegal_parameter" alert.

5.2. Authentication

When the "tls_cert_with_extern_psk" extension is successfully negotiated, authentication of the server depends upon the ability to generate a signature that can be validated with the public key in the server's certificate. This is accomplished by the server sending the Certificate and CertificateVerify messages as described in Sections 4.4.2 and 4.4.3 of [RFC8446].

TLS 1.3 does not permit the server to send a CertificateRequest message when a PSK is being used. This restriction is removed when the "tls_cert_with_extern_psk" extension is negotiated, allowing certificate-based authentication for both the client and the server. If certificate-based client authentication is desired, this is accomplished by the client sending the Certificate and CertificateVerify messages as described in Sections 4.4.2 and 4.4.3 of [RFC8446].

5.3. Keying Material

Section 7.1 of [RFC8446] specifies the TLS 1.3 Key Schedule. The successful negotiation of the "tls_cert_with_extern_psk" extension requires the key schedule processing to include both the external PSK and the (EC)DHE shared secret value.

If the client and the server have different values associated with the selected external PSK identifier, then the client and the server

will compute different values for every entry in the key schedule,
which will lead to the client aborting the handshake with a
"decrypt_error" alert.

6. IANA Considerations

IANA is requested to update the TLS ExtensionType Registry [IANA] to
include "tls_cert_with_extern_psk" with a value of TBD and the list
of messages "CH, SH" in which the "tls_cert_with_extern_psk"
extension may appear.

7. Security Considerations

The Security Considerations in [RFC8446] remain relevant.

TLS 1.3 [RFC8446] does not permit the server to send a
CertificateRequest message when a PSK is being used. This
restriction is removed when the "tls_cert_with_extern_psk" extension
is offered by the client and accepted by the server. However, TLS
1.3 does not permit an external PSK to be used in the same fashion as
a resumption PSK, and this extension does not alter those
restrictions. Thus, a certificate MUST NOT be used with a resumption
PSK.

Implementations must protect the external pre-shared key (PSK).
Compromise of the external PSK will make the encrypted session
content vulnerable to the future development of a large-scale quantum
computer. However, the generation, distribution, and management of
the external PSKs is out of scope for this specification.

Implementers should not transmit the same content on a connection
that is protected with an external PSK and a connection that is not.
Doing so may allow an eavesdropper to correlate the connections,
making the content vulnerable to the future invention of a large-
scale quantum computer.

Implementations must generate external PSKs with a secure key
management technique, such as pseudo-random generation of the key or
derivation of the key from one or more other secure keys. The use of
inadequate pseudo-random number generators (PRNGs) to generate
external PSKs can result in little or no security. An attacker may
find it much easier to reproduce the PRNG environment that produced
the external PSKs and search the resulting small set of
possibilities, rather than brute-force searching the whole key space.
The generation of quality random numbers is difficult. [RFC4086]
offers important guidance in this area.

If the external PSK is known to any party other than the client and
the server, then the external PSK MUST NOT be the sole basis for
authentication. The reasoning is explained in Section 4.2 of
[K2016]. When this extension is used, authentication is based on
certificates, not the external PSK.

In this extension, the external PSK preserves confidentiality if the
(EC)DH key agreement is ever broken by cryptanalysis or the future
invention of a large-scale quantum computer. As long as the attacker
does not know the PSK and the key derivation algorithm remains
unbroken, the attacker cannot derive the session secrets even if they
are able to compute the (EC)DH shared secret. Should the attacker be
able compute the (EC)DH shared secret, the forward secrecy advantages
traditionally associated with ephemeral (EC)DH keys will no longer be
relevant. Although the ephemeral private keys used during a given
TLS session are destroyed at the end of a session, preventing the
attacker from later accessing them, these private keys would
nevertheless be recoverable due to the break in the algorithm.
However, a more general notion of "secrecy after key material is
destroyed" would still be achievable using external PSKs, if they are
managed in a way that ensures their destruction when they are no
longer needed, and with the assumption that the algorithms that use
the external PSKs remain quantum-safe.

TLS 1.3 key derivation makes use of the HKDF algorithm, which depends
upon the HMAC [RFC2104] construction and a hash function. This
extension provides the desired protection for the session secrets as
long as HMAC with the selected hash function is a pseudorandom
function (PRF) [GGM1986].

This specification does not require that the external PSK is known
only by the client and server. The external PSK may be known to a
group. Since authentication depends on the public key in a
certificate, knowledge of the external PSK by other parties does not
enable impersonation. Since confidentiality depends on the shared
secret from (EC)DH, knowledge of the external PSK by other parties
does not enable eavesdropping. However, group members can record the
traffic of other members, and then decrypt it if they ever gain
access to a large-scale quantum computer. Also, when many parties
know the external PSK, there are many opportunities for theft of the
external PSK by an attacker. Once an attacker has the external PSK,
they can decrypt stored traffic if they ever gain access to a large-
scale quantum computer in the same manner as a legitimate group
member.

TLS 1.3 [RFC8446] takes a conservative approach to PSKs; they are
bound to a specific hash function and KDF. By contrast, TLS 1.2
[RFC5246] allows PSKs to be used with any hash function and the TLS

1.2 PRF. Thus, the safest approach is to use a PSK exclusively with
TLS 1.2 or exclusively with TLS 1.3. Given one PSK, one can derive a
PSK for exclusive use with TLS 1.2 and derive another PSK for
exclusive use with TLS 1.3 using the mechanism specified in
[I-D.ietf-tls-external-psk-importer].

TLS 1.3 [RFC8446] has received careful security analysis, and the
following informal reasoning shows that the addition of this
extension does not introduce any security defects. This extension
requires the use of certificates for authentication, but the
processing of certificates is unchanged by this extension. This
extension places an external PSK in the key schedule as part of the
computation of the Early Secret. In the initial handshake without
this extension, the Early Secret is computed as:

    Early Secret = HKDF-Extract(0, 0)

With this extension, the Early Secret is computed as:

    Early Secret = HKDF-Extract(External PSK, 0)

Any entropy contributed by the external PSK can only make the Early
Secret better; the External PSK cannot make it worse. For these two
reasons, TLS 1.3 continues to meet its security goals when this
extension is used.

8. Privacy Considerations

Appendix E.6 of [RFC8446] discusses identity exposure attacks on
PSKs. The guidance in this section remains relevant.

This extension makes use of external PSKs to improve resilience
against attackers that gain access to a large-scale quantum computer
in the future. This extension is always accompanied by the
"pre_shared_key" extension to provide the PSK identities in plaintext
in the ClientHello message. Passive observation of the these PSK
identities will aid an attacker to track users of this extension.

9. Acknowledgments

Many thanks to Liliya Akhmetzyanova, Roman Danyliw, Christian
Huitema, Ben Kaduk, Geoffrey Keating, Hugo Krawczyk, Mirja
Kuehlewind, Nikos Mavrogiannopoulos, Nick Sullivan, Martin Thomson,
and Peter Yee for their review and comments; their efforts have
improved this document.

10. References

10.1. 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,
           <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>.

[RFC8446] 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>.

10.2. Informative References

[DH1977]   Diffie, W. and M. Hellman, "New Directions in
           Cryptography", IEEE Transactions on Information
           Theory V.IT-22 n.6, June 1977.

[FIPS186] National Institute of Standards and Technology, "Digital
           Signature Standard (DSS)", Federal Information Processing
           Standards Publication (FIPS PUB) 186-4, July 2013.

[GGM1986] Goldreich, O., Goldwasser, S., and S. Micali, "How to
           construct random functions", J. ACM 1986 (33), pp.
           792-807, 1986.

[I-D.hoffman-c2pq]
           Hoffman, P., "The Transition from Classical to Post-
           Quantum Cryptography", draft-hoffman-c2pq-06 (work in
           progress), November 2019.

[I-D.ietf-tls-external-psk-importer]
           Benjamin, D. and C. Wood, "Importing External PSKs for
           TLS", draft-ietf-tls-external-psk-importer-02 (work in
           progress), November 2019.

[IANA]   "IANA Registry for TLS ExtensionType Values", n.d.,
           <https://www.iana.org/assignments/tls-extensiontype-
           values/tls-extensiontype-values.xhtml>.

   [IEEE1363]
             Institute of Electrical and Electronics Engineers, "IEEE
             Standard Specifications for Public-Key Cryptography", IEEE
             Std 1363-2000, 2000.

   [K2016]   Krawczyk, H., "A Unilateral-to-Mutual Authentication
             Compiler for Key Exchange (with Applications to Client
             Authentication in TLS 1.3)", IACR ePrint 2016/711, August
             2016.

   [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-
             Hashing for Message Authentication", RFC 2104,
             DOI 10.17487/RFC2104, February 1997,
             <https://www.rfc-editor.org/info/rfc2104>.

   [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker,
             "Randomness Requirements for Security", BCP 106, RFC 4086,
             DOI 10.17487/RFC4086, June 2005,
             <https://www.rfc-editor.org/info/rfc4086>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
             (TLS) Protocol Version 1.2", RFC 5246,
             DOI 10.17487/RFC5246, August 2008,
             <https://www.rfc-editor.org/info/rfc5246>.

   [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch,
             "PKCS #1: RSA Cryptography Specifications Version 2.2",
             RFC 8017, DOI 10.17487/RFC8017, November 2016,
             <https://www.rfc-editor.org/info/rfc8017>.

Author's Address

   Russ Housley
   Vigil Security, LLC
   516 Dranesville Road
   Herndon, VA 20170
   USA

   Email: housley@vigilsec.com