

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 30, 2019

R. Housley  
Vigil Security  
September 26, 2018

TLS 1.3 Extension for Certificate-based Authentication with an External  
Pre-Shared Key  
[draft-housley-tls-tls13-cert-with-extern-psk-02](#)

## Abstract

This document specifies a TLS 1.3 extension that allows a server to authenticate with a certificate while also providing a pre-shared key (PSK) as an input to the key schedule.

## 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 March 30, 2019.

## Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Internet-Draft

Certificate with External PSK

September 2018

## 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 either of these two types of PSK as an input to the TLS 1.3 key schedule.

## 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 motivation for using a certificate with an external PSK is different than the motivation for using a certificate with a resumption PSK.

### 3.1. Certificate With External PSK

The invention 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 and key agreement algorithm used to establish a pairwise shared secret between the client and server. 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 that

could be broken by the invention of a large-scale quantum computer [[I-D.hoffman-c2pq](#)]. These algorithms include Diffie-Hellman (DH) [[DH](#)] and Elliptic Curve Diffie-Hellman (ECDH) [[IEEE1363](#)]. 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.

When a certificate is used for authentication and a strong external PSK is used in conjunction with a key agreement algorithm, today's communications can be protected from the future invention of a large-scale quantum computer. The strong external PSK and the shared secret from the key agreement algorithms are both provided as inputs to the TLS 1.3 key schedule, which preserves the authentication provided by the existing certificate and digital signature mechanisms, and requires the attacker to learn the external PSK as well as the shared secret to break confidentiality.

### [3.2.](#) Certificate With Resumption PSK

There are two motivations for using a certificate with a resumption PSK.

In the first situation, the client seeks corroboration that the server has access to the private key associated with the certificate. That is, the server uses the same certificate in this handshake as was used to establish the resumption PSK. Successful completion of the handshake requires the server to produce a valid signature in the CertificateVerify handshake message.

In the second situation, the server wishes to use a different certificate for the resumption handshake, which allows the resumed session to be associated with a different server identity. Successful completion of the handshake requires the server to produce a valid signature in the CertificateVerify message that can be validated with the public key in the certificate that is provided in the Certificate handshake message.

### [3.3.](#) Design Considerations With Early Data

When a client provides early data and makes use of a certificate with

a resumption PSK, the server MUST use the same certificate, public key, and private key as in the original handshake. Doing otherwise would create an ambiguity about the server identity that received the early data. For this reason, the handshake fails if the client sends early data and the server uses a different certificate with a resumption PSK.

#### 4. Extension Overview

This section provides a brief overview of the "tls\_cert\_with\_psk" extension.

The client includes the "tls\_cert\_with\_psk" extension in the ClientHello message. The "tls\_cert\_with\_psk" extension MUST be accompanied by the "key\_share", "psk\_key\_exchange\_modes", and "pre\_shared\_key" extensions. The "pre\_shared\_key" extension MUST be the last extension in the ClientHello message, and it provides a list of PSK identifiers that the client is willing to use with this server. If the "tls\_cert\_with\_psk" extension is used with a resumption PSK and the "early\_data" extension, then the client MUST check that the server provided the same certificate as was used in the initial handshake. These extensions are all described in [Section 4.2 of \[RFC8446\]](#).

If the server is willing to use one of the PSKs listed in the "pre\_shared\_key" extension and perform certificate-based authentication, then the server includes the "tls\_cert\_with\_psk" extension in the ServerHello message. The "tls\_cert\_with\_psk" extension MUST be accompanied by the "key\_share" and "pre\_shared\_key" extensions. If none of the PSKs in the list provided by the client is acceptable to the server, then the "tls\_cert\_with\_psk" extension is omitted from the ServerHello message.

The successful negotiation of the "tls\_cert\_with\_psk" extension requires the TLS 1.3 key schedule processing to include both the selected 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 PSK.

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 PSK.

As required by [Section 4.2.11 of \[RFC8446\]](#), each external PSK is associated with a single Hash algorithm. The hash algorithm MUST be set when the external PSK is established, with a default of SHA-256 if no hash algorithm is specified during establishment.

Resumption PSKs are established via the ticket mechanism described in [Section 4.6.1 of \[RFC8446\]](#). The hash algorithm associated with the resumption PSK MUST be the same KDF hash algorithm as that used to establish the initial session. This is the KDF hash algorithm of the session where the ticket was established.

## [5.](#) Certificate with PSK Extension

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

The TLS 1.3 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 \[RFC8446\]](#), and it MUST include the same "tls\_cert\_with\_psk" extension as the original ClientHello message or abort the handshake.

Many server extensions are carried in the EncryptedExtensions message; however, the "tls\_cert\_with\_psk" extension is carried in the

ServerHello message. It is only present in the ServerHello message if the server recognizes the "tls\_cert\_with\_psk" extension and the server possesses one of the 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\_psk" extension, adding one new type to ExtensionType:

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

In an initial handshake, the "tls\_cert\_with\_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. In a subsequent handshake, the "tls\_cert\_with\_psk" extension is relevant when the client and server possess a resumptions PSK in common and server authentication with a certificate is desired. The "tls\_cert\_with\_psk" extension has the following syntax:

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

To use a PSK with certificates, clients MUST provide the "tls\_cert\_with\_psk" extension, and it MUST be accompanied by the "key\_share", "psk\_key\_exchange\_modes", and "pre\_shared\_key" extensions in the ClientHello. If clients offer a "tls\_cert\_with\_psk" extension without all of these other extensions, servers MUST abort the handshake. The client MAY also find it useful to include the the "supported\_groups" extension. If clients offer a "early\_data" extension during a resumption handshake, then clients MUST confirm that the server uses the same certificate, public key, and private key as in the handshake that established the resumption PSK. Note that [Section 4.2 of \[RFC8446\]](#) allows extensions to appear in any order, with the exception of the "pre\_shared\_key" extension, which MUST be the last extension in the ClientHello. Also, there MUST NOT be more than one instance of any extension in the ClientHello message.

The "key\_share" extension is defined in [Section 4.2.8 of \[RFC8446\]](#).

The "psk\_key\_exchange\_modes" extension is defined in [Section 4.2.9 of \[RFC8446\]](#). The "psk\_key\_exchange\_modes" extension restricts both the use of PSKs offered in this ClientHello and those which the server might supply via a subsequent NewSessionTicket. As a result, clients MUST include the psk\_dhe\_ke mode for an initial handshake, and 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 "early\_data" extension is defined in [Section 4.2.10 of \[RFC8446\]](#).

The "supported\_groups" extension is defined in [Section 4.2.7 of \[RFC8446\]](#).

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.

```

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>;
} OfferedPsk;

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

```

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

The identities are a list of PSK identities that the client is willing to negotiate with the server. Each PSK has an associated identity that is known to the client and the server. (The identity is also referred to as an identifier or a label.)

The obfuscated\_ticket\_age is not used for external PSKs; clients SHOULD set this value to 0, and servers MUST ignore the value. The obfuscated\_ticket\_age is used for resumption PSKs, and [Section 4.2.11.1 of \[RFC8446\]](#) describes how to form this value for identities established via the NewSessionTicket message.

The binders are a series of HMAC values, one for each 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 PSK, and a partial transcript of the current handshake. Generation of the binder\_key from the 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 PSK identity that the server selected from the list offered by the client. If none of the offered PSKs in the list provided by the client are acceptable to the server, then the "tls\_cert\_with\_psk" extension MUST be omitted from the ServerHello message. The server MUST validate the binder value that corresponds to the selected PSK as described in [Section 4.2.11.2 of \[RFC8446\]](#). If the binder does not validate, the server MUST abort the handshake with an "illegal\_parameter" alert. Servers SHOULD NOT attempt to validate multiple binders; rather they SHOULD select one of the offered PSKs and validate only the binder that corresponds to that PSK.

When the "tls\_cert\_with\_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\_psk" extension is negotiated, allowing the 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 [\[RFC8446\]](#).

[Section 7.1 of \[RFC8446\]](#) specifies the TLS 1.3 Key Schedule. The successful negotiation of the "tls\_cert\_with\_psk" extension requires the key schedule processing in the initial handshake to include both the external PSK and the (EC)DHE shared secret value. In a resumption handshake, the resumption PSK MUST be used in the key schedule, and the (EC)DHE shared secret MAY also be used.

If the client and the server have different values associated with the selected PSK identifier, then the client and the server will compute different values for every entry in the TLS 1.3 key schedule, which will lead to the termination of the connection with a "decrypt\_error" alert.

## 6. IANA Considerations

IANA is requested to update the TLS ExtensionType Registry to include "tls\_cert\_with\_psk" with a value of TBD and the list of messages "CH, SH" in which the "tls\_cert\_with\_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\_psk" extension is offered by the client and accepted by the server.

Implementations need to protect the pre-shared key (PSK). Compromise of the external PSK used in the initial handshake makes the encrypted session content vulnerable to the future invention of a large-scale quantum computer. Compromise of the resumption PSK makes the encrypted session content associated with subsequent sessions vulnerable to an attacker that knows the PSK, and it allows the attacker to initiate new sessions which are also vulnerable.

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.

Deployment of a pairwise external PSK between every client and server is not practical. Instead, this specification envisions an external PSK being distributed to a group of clients and group of servers. At some point in the future a large-scale quantum computer might get invented, and if any member of the group has access to it, then that group member can recover the traffic associated with the PSK. However, parties outside the group cannot recover the traffic because the large-scale quantum computer does not assist with the discovery of the external PSK of reasonable size.

Implementations must choose 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 searching 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.

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 with either TLS 1.2 or TLS 1.3. However, any PSK that might be used with both TLS 1.2 and TLS 1.3 must be used with only one hash function, which is the one that is bound for use in TLS 1.3. This restriction is less than optimal when users want to provision a single PSK. While the constructions used in TLS 1.2 and TLS 1.3 are both based on HMAC [RFC2104], the constructions are different, and there is no known way in which reuse of the same PSK in TLS 1.2 and TLS 1.3 that would produce related outputs.

## 8. Acknowledgments

Many thanks to Nikos Mavrogiannopoulos, Nick Sullivan, Martin Thomson, and Peter Yee for their review and comments; their efforts have improved this document.

## 9. References

### 9.1. 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>>.

## 9.2. Informative References

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

Housley

Expires March 30, 2019

[Page 10]

---

Internet-Draft

Certificate with External PSK

September 2018

[I-D.hoffman-c2pq]

Hoffman, P., "The Transition from Classical to Post-Quantum Cryptography", [draft-hoffman-c2pq-04](#) (work in progress), August 2018.

[IEEE1363]

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

[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>>.

### Author's Address

Russ Housley  
Vigil Security, LLC  
918 Spring Knoll Drive  
Herndon, VA 20170  
USA

Email: [housley@vigilsec.com](mailto:housley@vigilsec.com)

Housley

Expires March 30, 2019

[Page 11]