

TLS 1.3 Extension for Certificate-based Authentication with an External Pre-Shared Key

draft-ietf-tls-tls13-cert-with-extern-psk
(Experimental)

Russ Housley
TLS WG at IETF 104
March 2019

Reminder: This Extension Adds Another Choice for the Initial Handshake

Initial Handshake:

Authentication:

Signature and Certificate

Signature and Certificate

Key Schedule Secret Inputs:

(EC)DHE

External PSK + (EC)DHE

Subsequent Handshake:

Authentication:

Resumption PSK

Resumption PSK

Key Schedule Secret Inputs:

Resumption PSK + (EC)DHE

(EC)DHE

Reminder: External PSK in Initial Handshake

Client

```
ClientHello
+ tls_cert_with_psk
+ supported_groups*
+ key_share
+ signature_algorithms*
+ psk_key_exchange_modes(psk_dhe_ke)
+ pre_shared_key
```

----->

```
{Certificate*}
{CertificateVerify*}
{Finished}
[Application Data]
```

Server

```
ServerHello
+ tls_cert_with_psk
+ key_share
+ pre_shared_key
+ {EncryptedExtensions}
{CertificateRequest*}
{Certificate}
{CertificateVerify}
{Finished}
```

<-----

----->

<----->

[Application Data]

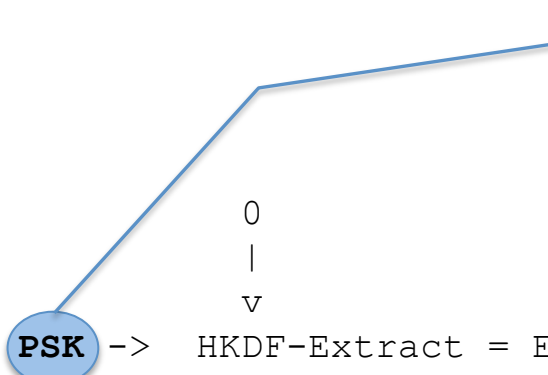
Extension Syntax

- The successful negotiation of the "tls_cert_with_extern_psk" extension requires the TLS 1.3 key schedule processing to include *both* the selected external PSK and the (EC)DHE shared secret value; it also requires the server to send the Certificate and CertificateVerify messages in the handshake
- The "tls_cert_with_extern_psk" extension is always be used along with the already defined "key_share", "psk_key_exchange_modes", and "pre_shared_key" extensions
- The "psk_key_exchange_modes" extension will always offer psk_dhe_ke
- The "pre_shared_key" extension used with obfuscated_ticket_age of zero, and only offer external PSKs
- Inclusion of the extension is willingness to authenticate the server with a certificate and include an external PSK in the key schedule processing:

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

Key Schedule

The external PSK is placed in the usual place in the key schedule. In the initial handshake, the PSK input value is otherwise set to 0.



```
0
|
v
PSK -> HKDF-Extract = Early Secret
      |
      +-----> Derive-Secret(., "ext binder" | "res binder", "")
                = binder_key
      |
      +-----> Derive-Secret(., "c e traffic", ClientHello)
                = client_early_traffic_secret
      |
      +-----> Derive-Secret(., "e exp master", ClientHello)
                = early_exporter_master_secret
      v
Derive-Secret(., "derived", "")
.
.
.
```

Informal Reasoning About TLS 1.3 Security

Authentication

The certificate processing is exactly the same. It is not better or worse. No changes.

Key Schedule computation of Early Secret

- Initial Handshake

 - Without extension: $\text{HKDF-Extract}(0, 0)$

 - With extension: $\text{HKDF-Extract}(\text{ExternalPSK}, 0)$

- Subsequent Handshake

 - No changes.

Conclusion: Any entropy contributed by the External PSK can only make the Early Secret better; the External PSK cannot make it worse.

Next Steps

- We have been talking about this topic since IETF 101
- Along the way, a bunch of stuff was added, and then it was removed again
- We are essentially back to the original -00 document
- I think the document is ready for WG Last Call for progression as an Experimental document

- PLEASE REVIEW!