draft-rescorla-tls13-semistatic-dh-00

Eric Rescorla
Nick Sullivan

Overview

- TLS 1.3 key schedule is similar to OPTLS 1.3 [1], supporting:
  - 1-RTT
  - PSK-based resumption
  - (EC)DHE+PKE-based resumption
- Missing: **1-RTT semi-static** mode
  - Server supplies signed semi-static key share (in a delegated credential or certificate)
  - Peers generate static secret from client ephemeral and server semi-static key shares
  - Replace `CertificateVerify` with a MAC computed from the static secret

[1] https://eprint.iacr.org/2015/978.pdf

Motivation

- Resumption PSKs can be known to the server without the session ticket
  - 0-RTT is only possible with either external PSKs or resumption PSKs
  - There is some interest in "anonymous" 0-RTT -- server can’t link new session to old
  - Semi-static DH allows the set of clients to share the same anonymity set
- Single-primitive protocols
  - Some implementations have code size restrictions
  - Some clients have an awkward boundary with PKI validation
  - Server only needs Diffie-Hellman, client can get by with DH-only + key pinning
- Allows client to mix ephemeral key into early data secret
  - Heavy 0-RTT uses of TLS are not forward-secret, can we add fresh entropy into 0-RTT?
- Mixes long-term server key into master secret.
  - Protects against compromise of server PRNG.

Motivation (cont’d)

- Delegated credentials makes this possible without ed25519 certificates
  - Time-bounded key swap [EKR: what is this?]
- Servers with RSA certificates may want to limit RSA operations
  - Delegated x25519 credential reduces computation cost
- There are scenarios where semi-static key distribution makes sense
  - As replacement for pre-shared PSK modes (in IoT)
  - Fetching from a central repository Certificate Transparency log or other
  - Fetching from .well-known (??)
- OPTLS has nice properties and Karthik promised it would come back one day

TLS 1.3 Non-Semi-Static Key Schedule

```
...
          Derive-Secret(., "derived", "")
                |
                v
     0 -> HKDF-Extract = Master Secret
                |
                +-----> Derive-Secret(., "c ap traffic",
                |                     ClientHello...server Finished)
                |                     =
client_application_traffic_secret_0
                |
...
```

TLS 1.3 Semi-Static Key Schedule

```
...
           Derive-Secret(., "derived", "")
                 |
                 v
     SS -> HKDF-Extract = Master Secret
                 |
                 +-----> Derive-Secret(., "c ap traffic",
                 |                     ClientHello...server Finished)
                 |                     =
client_application_traffic_secret_0
                 |
...
```

Negotiation Details

- Use a new signature scheme

```
enum {
   sig_p256(0x0901),
   sig_p384(0x0902),
   sig_p521(0x0903),
   sig_x52219(0x0904),
   sig_x448(0x0905),
 } SignatureScheme;
```

- These values **MUST NOT** appear in "signature_algorithms_cert"

Open Questions

- Should we keep a CertificateVerify message
- Can client authentication happen the same way?
- Diffie-Hellman group must support multiple uses of the same key
  - Complicates some post-quantum schemes (SIDH)
- Should the semi-static key part of the key schedule or only the CertificateVerify?
  - Analysis is simpler if used for MAC only
  - Analogy to signature-based flow is more direct

Early Data

https://github.com/ekr/draft-rescorla-tls13-semistatic-dh/pull/3

```
                0
                |
                v
ESS ->  HKDF-Extract = Early Ephemeral Secret (ignored)
                |
                +
        Derive-Secret(., "derived", "")
                |
                v
PSK ->  HKDF-Extract = Early Secret
                |
               ...
```

Open Questions about semi-static (re)use

- Need a public key operation to read early data (DoS potential?)
- How to choose the PSK identity
    - Hash of secret?
    - String corresponding to values that would be stored in the ticket (ALPN, Cipher, etc.)
- Should the server be able to send a different semi-static key than the one used in the handshake?
    - Raw DH key?
    - Opaque session ticket?

Questions for the working group

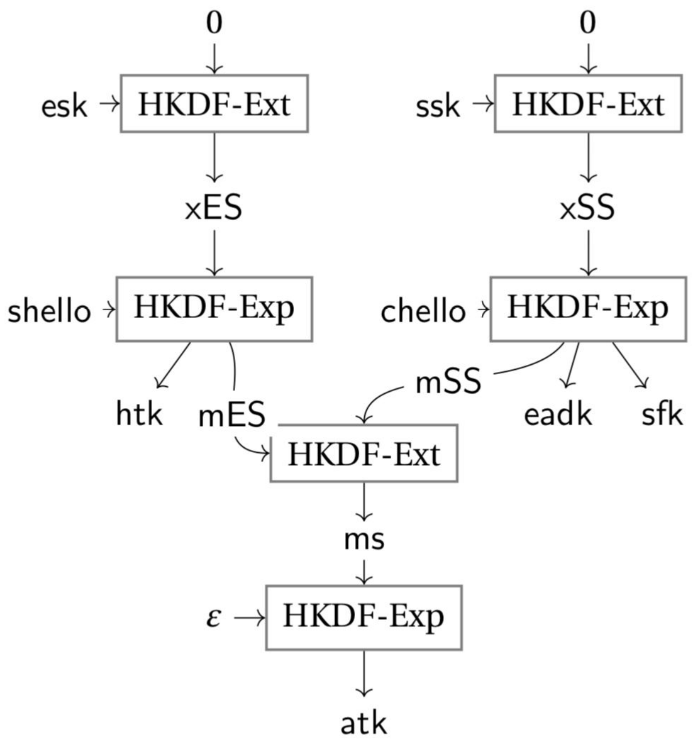
Is this work that people are interested in pursuing?

Should extensions (signature schemes in this case) be able to change the handshake so dramatically?

Are there people willing to review the draft?

Backup Slides

OPTLS 1.3 Key Schedule



1-RTT semi-static	ssk = g^{xs}, esk = g^{xy}
1-RTT non-static	ssk = esk = g^{xy}
PSK	ssk = esk = psk
PSK-DHE	ssk = psk, esk = g^{xy}