TLS 1.3

draft-ietf-tls-tls13-14

Eric Rescorla
Mozilla
ekr@rtfm.com
Major changes since draft-12

- Remove 0-RTT (EC)DHE and client auth *
- Complete 0-RTT PSK mode *
- Restructure key schedule *
- Add session context *
- Fully define HelloRetryRequest *
- NewSession ticket use flags
- Allow server to send SupportedGroups
- Move CertificateStatus to an extension
- Add ticket age for anti-replay
- Allow resumption after fatal alerts
- Remove non-closure warning alerts
- Add Security Analysis section
0-RTT is now PSK-only

ClientHello
+ early_data
+ pre_shared_key
+ key_share*
(Finished)
(Application Data*)
(end_of_early_data) -------->

ServerHello
+ early_data
+ pre_shared_key
+ key_share*
{EncryptedExtensions}
{CertificateRequest*}
{Finished}
<-------- [Application Data*]

{Certificate*}
{CertificateVerify*}
{Finished} -------->

[Application Data] <--------> [Application Data]
PSK $\rightarrow$ HKDF-Extract

$\rightarrow$ Derive-Secret(., "early traffic secret", ClientHello)

$=$ early_traffic_secret

(ECDHE) $\rightarrow$ HKDF-Extract

Handshake

Secret $\rightarrow$ Derive-Secret(., "handshake traffic secret", ClientHello + ServerHello)

$=$ handshake_traffic_secret

0 $\rightarrow$ HKDF-Extract

Master Secret

$\rightarrow$ Derive-Secret(., "application traffic secret", ClientHello...Server Finished)

$=$ traffic_secret_0

$\rightarrow$ Derive-Secret(., "exporter master secret", ClientHello...Client Finished)

$=$ exporter_secret

$\rightarrow$ Derive-Secret(., "resumption master secret", ClientHello...Client Finished)

$=$ resumption_secret
Session Context

- Multiple requests to include more context when resuming (Krawczyk, Bhargavan)
  
  \[\text{resumption}_\text{psk} = \text{HKDF-Expand-Label}(\text{resumption}_\text{secret},
  "\text{resumption psk}"', "", L)\]

  \[\text{resumption}_\text{context} = \text{HKDF-Expand-Label}(\text{resumption}_\text{secret},
  "\text{resumption context}"', "", L)\]

- Merged into handshake hashes whenever used

  \[\text{Hash(}\text{Messages}) + \text{Hash(}\text{resumption}_\text{context})\]
Cookies for HelloRetryRequest

- Derived from DTLS (and originally Photuris)
- Server can provide a cookie with HRR
- Client echoes it with new ClientHello
- Usable for stateless reject by pickling the handshake state in the cookie
Post-Handshake Key Separation

- General consensus on list to leave as-is
- Analysis from Hugo Krawczyk indicates this is OK
- IMPORTANT: We still have key separation for ordinary-handshake and app data
Cipher Suite Negotiation: Problem Statement

- The cipher suite negotiation has gotten clunky and non-orthogonal

- Already was bad in 1.2
  - Cipher suite, signature algorithms, named groups

- Worse in 1.3
  - PSK, key shares

- Can we radically simplify?
Cipher Suite Negotiation: Overview

- Break up into the following axes
  - AEAD-PRF
  - Signature algorithms
  - Key shares/named groups
  - PSK

- Negotiate each separately
  - Straightforward for public key
  - PSK makes things a bit complicated
Public key algorithm negotiation

- Cipher suite just indicates AEAD and PRF
  - Probably define new cipher suites
  - Added bonus of letting us prune!

- Signature algorithms determines server cert/key and signature scheme

- Key shares and supported groups determine the key exchange
  - Model everything as (EC)DHE
  - Server’s key share indicates which group it picked
What about PSK?

- PSK can be combined with (EC)DHE and signatures (new) (？)

```c
enum { psk_ke(0), psk_dhe_ke(1), (255) } PskKeModes;
enum { psk_auth(0), psk_sign_auth(1), (255) } PskAuthModes;

struct {
    PskAuthMode auth_modes<1..255>
    PskKeMode ke_modes<1..255>
    opaque identity<0..2^16-1>
} PskIdentity;

struct {
    select (Role) {
        case client:
            PskIdentity identities<2..2^16-1>
        case server:
            PskAuthMode auth_mode;
            PskKeMode ke_mode;
            uint16 selected_identity;
    }
} PreSharedKeyExtension;
```
Should we change negotiation?

- **Cons**
  - Big change at the last minute
  - Makes APIs more complicated because the cipher suite doesn’t tell you everything
  - Doesn’t let you express non-orthogonal options

- **Pros**
  - Much easier to implement (based on initial prototypes)
  - Removes odd pairing of (EC)DHE and PSK cipher suites
  - More expressive

- **Proposal:** provisionally adopt pending a PR
Version Negotiation

ONE DOES NOT SIMPLY

INCREASE THE TLS VERSION NUMBER
Alternate Proposal

- Keep ClientHello version number at 3, 3 (TLS 1.2)
- Introduce a new tls_version extension
  - Semantic is: a list of all supported versions
  - Example: [ [3, 2], [3, 3], [3, 4], [53, 100] ]
- ServerHello contains the negotiated version
- All future versions negotiated this way
  - Can fuzz for futureproofing
- Discuss
PSK and Client Auth

• Draft implies support for client authentication even with PSK mode
  – Server just sends CertificateRequest
  – Semantics of this are odd.
  – 0-RTT is even worse
• Main proposal
  – CertificateRequest not allowed when using PSK
  – Use post-handshake client auth if you want this
• Fallback proposal
  – PSK client auth needs an identity that is “morally the same”
  – Then clients can refuse to refresh
• Proposed resolution: ban client auth PSK
Resumptions Contexts and 0-RTT Finished

- From the 0-RTT Finished:
  - Proof of at least partial liveness of the PSK [via ticket age]
  - An integrity check for the information in the ClientHello
- From the resumption context:
  - Tie the context from the PSK-establishing connection to future handshakes.
- Issues
  - “0” resumption context for out-of-band PSK is problematic
  - This seems duplicative
  - Reading the 0-RTT Finished is kind of a pain
  - Always adding the PSK context to the hash is clunky
Potential Options

• Remove 0-RTT Finished but use it as resumption_ctx
  - resumption_ctx = HMAC(. , ClientHello)

• Always require 0-RTT Finished even w/o 0-RTT (and include in the log)

• Always include a special Finished extension when using PSK
  - And discard resumption_ctx
  - This can be a bit tricky to implement

• Do nothing

• Proposal: ???
Crypto for Embedded 0-RTT Finished (thanks to Antoine)

Early Secret = HKDF-Extract(0, PSK)

early_finished_secret =
    Derive-Secret(Early Secret, "...", ClientHello-prefix)

ClientHello = ClientHello-prefix + HMAC(efs, ClientHello-prefix)

early_traffic_secret =
    Derive-Secret(Early Secret, "...", ClientHello)

Alternate, crazy idea:

ClientHello = ClientHello-prefix + AEAD(efs,
                                           ClientHello-prefix,
                                           <stuff>)
Multiple Concurrent Tickets (PR #8)

- Currently we implicitly support multiple tickets
  - Useful for de-linkage privacy, etc.
- Ticket encoding gives no guidance about how to use them
  - Is ticket $N$ usable after I see ticket $N + 1$? Try it and see!
- Proposal: Add a field (generation?) to indicate whether a ticket supersedes others
Last-minute thought: EE in Second Flight

• Should we put an extensions block in client’s second flight?
  • Pro
    – Only place to put encrypted data from client
    – We might really want this later
  • Con
    – Unspecified semantics
    – Not included in HS transcript
# Interop Status

<table>
<thead>
<tr>
<th>draft-ietf-tls-tls13-13 interop</th>
<th>NSS</th>
<th>BoringSSL</th>
<th>miTLS</th>
<th>ProtoTLS</th>
<th>mint</th>
<th>BoGo</th>
<th>TLS-tris</th>
</tr>
</thead>
<tbody>
<tr>
<td>client ↓</td>
<td>server →</td>
<td>1RZC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1RZ</td>
<td>1</td>
</tr>
<tr>
<td>NSS</td>
<td></td>
<td>1</td>
<td>1CH</td>
<td>1</td>
<td>1</td>
<td>1CH</td>
<td>1</td>
</tr>
<tr>
<td>BoringSSL</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>miTLS</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ProtoTLS</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>mint</td>
<td></td>
<td>1RZ</td>
<td>@svaldez</td>
<td>1</td>
<td>1RZ</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>BoGo</td>
<td></td>
<td>1</td>
<td>1CH</td>
<td>1</td>
<td>1</td>
<td>1H</td>
<td>@nharper</td>
</tr>
<tr>
<td>TLS-tris</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

- **self-test**
- **interop**
- **known broken**
- **unknown**
- **N/A**

**To Test:**

- **1=1-RTT**
- **R=Resumption**
- **Z=0-RTT**
- **C=Client Auth**
- **K=KeyUpdate**
- **H=HelloRetryRequest**
Timeline: Option #1 (No big changes)

Aug 8th     draft-15: Wire format frozen ("Cryptographer's version")
Aug 22nd    Implementations of draft-15
Aug 29th    draft-16: Revised based on feedback
Aug 29th    WGLC
Sep 30th    WGLC Ends
### Timeline: Option #2 (Change Negotiation or 0-RTT Finished)

<table>
<thead>
<tr>
<th>Date</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug 8th</td>
<td>draft-15: Changes agreed at IETF 96</td>
</tr>
<tr>
<td>Aug 22nd</td>
<td>Implementations of draft-15</td>
</tr>
<tr>
<td>Aug 29th</td>
<td>draft-16: Revised; Wire format frozen (&quot;cryptographer’s version&quot;)</td>
</tr>
<tr>
<td>Sep 12th</td>
<td>Implementations of draft-16</td>
</tr>
<tr>
<td>Sep 19th</td>
<td>draft-17: Revised based on feedback</td>
</tr>
<tr>
<td>Sep 19th</td>
<td>WGLC</td>
</tr>
<tr>
<td>Oct 17th</td>
<td>WGLC Ends</td>
</tr>
</tbody>
</table>