

TCP-use-TLS

Eric Rescorla

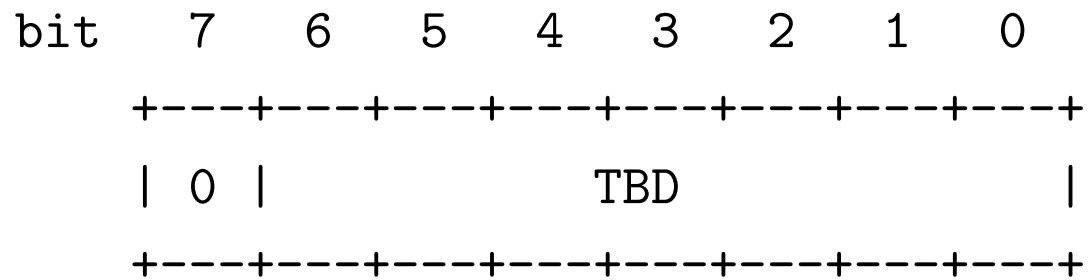
Mozilla

`ekr@rtfm.com`

Basic idea

- Use ENO to negotiate use of the `_spec_` defined in this draft
- Run TLS as usual over TCP once ENO negotiates it
- Profile TLS down to a small subset of TLS 1.3
 - But also allow TLS 1.2 (see below)
- Intuition: if you ignore ENO extensions this looks just like TLS-over-TCP

Minimal SYN Suboption



- As usual, application-aware bit avoids TLS-in-TLS

SYN/ACK Suboptions

```
One-byte   +-----+
1-RTT      |   TBD   |
Only       +-----+
```

```
Variable   +-----+-----//-----+
1-RTT or   |1| TBD |           Nonce   |
0-RTT      +-----+-----//-----+
```

- Warning: nonce thing may be half-baked; more on this later

TLS 1.3 Profile Overview

- ECDHE only (MUST P256, SHOULD X25519)
- Limited set of symmetric ciphers
- Support for raw public keys (avoid need for X.509 certs)
- Server “authentication” mandatory for protocol reasons
- No client authentication
- No resumption

TLS 1.3 Handshake in TCPINC (1-RTT)

```
SYN + TCP-ENO [TLS]          ----->
                               <----- SYN/ACK + TCP-ENO [TLS]
ACK + TCP-ENO                ----->
ClientHello
  + ClientKeyShare
  + TCPENOTranscript         ----->
                               ServerHello
                               ServerKeyShare
                               {EncryptedExtensions}
                               {ServerConfiguration*}
                               {Certificate}
                               {CertificateVerify}
                               <----- {Finished}
                               <----- [Application Data]
{Finished}                   ----->
[Application Data]           <-----> [Application Data]
```

0-RTT

- In initial handshake, server provides long-term ECDHE key in ServerConfiguration
- In subsequent handshake, client can encrypt using that key
 - Obviously this doesn't provide PFS
 - There are replay issues in stock TLS 1.3
 - ... but here we have a nonce
- 0-RTT currently required to be manually configured

TLS 1.3 Handshake in TCPINC (0-RTT)

```
SYN + TCP-ENO [TLS]      ----->
                          <-----
                          SYN/ACK + TCP-ENO
                          [TLS + Nonce]

ACK + TCP-ENO           ----->
ClientHello
+ ClientKeyShare
+ EarlyDataIndication
+ TCPENOTranscript
(EncryptedExtensions)
(Finished)
(Application Data)      ----->

                          ServerHello
                          + EarlyDataIndication
                          ServerKeyShare
                          {EncryptedExtensions}
                          {ServerConfiguration*}
                          {Certificate}
                          {CertificateVerify}
                          {Finished}
                          [Application Data]
                          <-----
                          <-----
{Finished}              ----->
[Application Data]     <-----> [Application Data]
```


Session IDs

- Computed as TLS Exporter [RFC 5705]
- Label to exporter is TBD
- TCP-ENO transcript bound in via TCPENOTranscript extension

Open issues

- Ability to negotiate TLS 1.2 as well
 - This is useful, though perhaps not for TCPINC use case
 - Some suggestions to define two code points
- Should the client be able to say it wants 0-RTT?
 - How would this help?
- Interaction with TFO

Questions?