Public Service Announcement: DTLS 1.3

`draft-rescorla-tls-dtls13-00`

Eric Rescorla

Mozilla

ekr@rtfm.com

Most of this work by Hannes Tschofenig

Overview

- DTLS version of TLS 1.3
- Still presented as a delta from TLS 1.3
- Some improvements/cleanup
- Partly informed by early implementation experience

Message flows

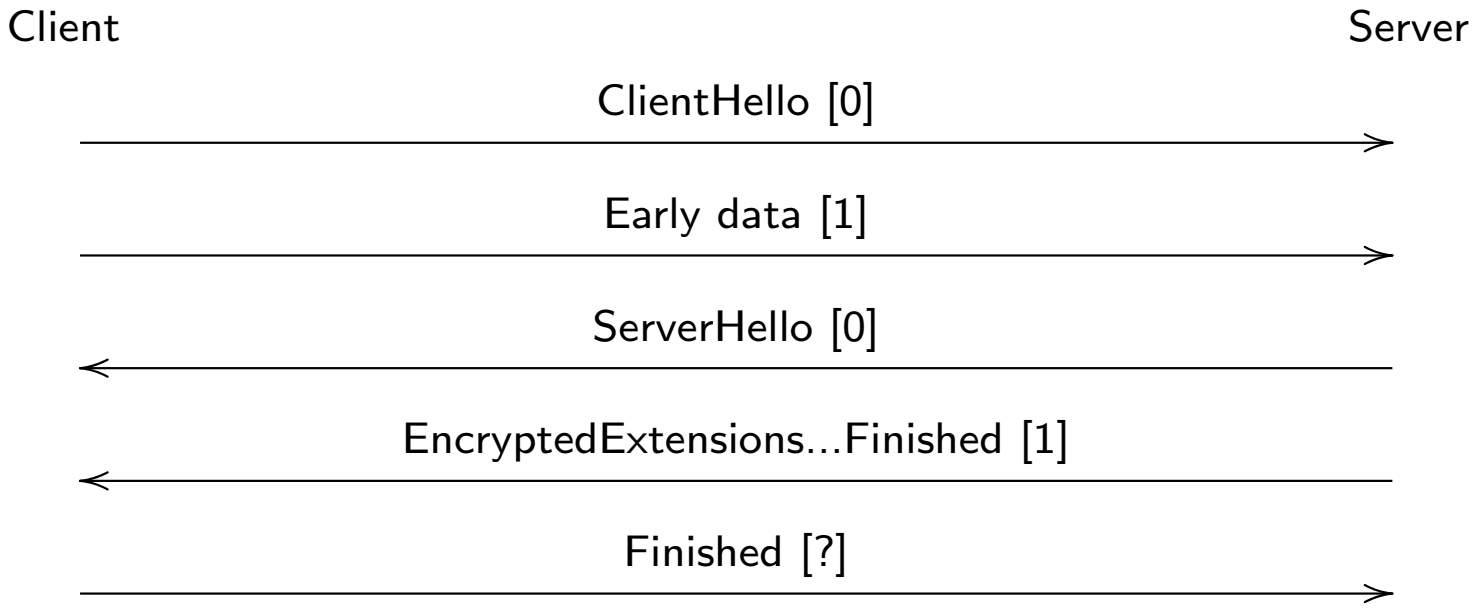
- Mostly just adopt the TLS 1.3 message flows
- Deprecate HelloVerifyRequest in favor of HelloRetryRequest
- This was always our plan

ACKs

- DTLS historically used an implicit ACK
  - Receiving the start of the next flight means the flight was received
- Simple (but also simpleminded)
  - Slightly tricky to implement
  - Gives limited congestion feedback
  - Handles single-packet loss badly
- Interacts badly with some TLS 1.3 features (like NST)
- Solution: introduce an explicit ACK
  - Details TBD (should this include timestamp, SACK, fragments, etc.)

Epochs

- DTLS uses epochs to indicate key changes (incrementing by one)
- But this causes confusion with 0-RTT



- Solution: use fixed epochs for HS flights (specific values TBD, draft is wrong)

KeyUpdate

- KeyUpdate might get lost (though see ACK)
- KeyUpdate seems redundant with epochs
  - Just update keys when you receive a new epoch from the other side
- Can be tricky to implement
  - Need to successfully decrypt before updating
  - What if attacker gives you an update far in the future?
  - How many updates can oyu have outstanding
- Some open issues here

Some other open issues

- Which version numbers to use on the wire (hint: regular TLS 1.3 numbers)
- Should we reduce DTLS record header size?
  - Strip out fixed first three bytes
  - Shorten the sequence number
- Do we want a connection ID?
  - Most frequently asked for DTLS feature
- ???

Next steps

- draft-01 to update to TLS 1.3 draft-18
- Update NSS (and maybe mBed) implementations to validate
- Ask for WG acceptance
- ???
- Profit