DTLS 1.3

draft-ietf-tls-dtls13-01

Eric Rescorla    Hannes Tschofenig    Nagendra Modadugu
Mozilla          ARM                  Google
Reminder: 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
Current proposal: SACK

- ACKs contain the sequence numbers of received records
  - From the current flight only
  - Senders need to maintain a map from records to handshake messages
  - Senders SHOULD NOT retransmit ACKed data and MUST NOT retransmit ACKed flights

- Separate record type, not a handshake record
  - MUST be sent with epoch $\geq$ than what’s being ACKed
  - Sent with the current sending key

- Receiving the next flight is an implicit ACK
When should receivers ACK

• When receiving messages that don’t have in-handshake responses
• When it looks like messages might have gotten lost
  – When you get an out-of-order record
  – When you get a partial record and don’t get the rest “immediately”
• Not for non-handshake messages
Reduced Headers

- What can we remove?
  - Nonce
  - Content type and version (hopefully)

- Proposal (thanks to MT):

```c
struct {
    uint16 epoch_sequence // format = 001eesss ssssssss
    uint16 length;
    opaque encrypted_record[length];
} DtlsHeader;
```
Connection IDs

- Lack of Connection IDs clearly a problem for NATs/IoT, etc.
- Connection IDs are also a clear privacy problem
  - Lots of proposals for how to do privacy preserving Conn IDs
  - ... but they’re complicated and none of them seem totally baked
  - This seems like less of a privacy problem than with browsers (QUIC)
- Proposal: use a fixed connection ID for now
  - In an extension
  - We can always replace it later
Concrete proposal

struct {
    opaque connection_id<0..255>;
} ConnectionId;

struct {
    uint16 epoch_sequence // format = 001eesss ssssssss
    opaque connection_id[connection_id_length];
    uint16 length;
    opaque encrypted_record[length];
} DtlsHeader;

• IDs are used if client offers and server answers
  – On all (non-0RTT)? encrypted records
• Each side sends with the other’s ID
  – Because IDs are unframed, 0-length IDs are just omitted
Other issues?