Changes since -01

- Short record headers
- Empty ACK and clarified ACK rules
- Reintroduce KeyUpdate because it now works with ACKs
Short headers 1: Shorten DTLS\texttt{Ciphertext}

```c
struct {
    ContentType opaque_type = 23; /* application_data */
    uint32 epoch_and_sequence;
    uint16 length;
    opaque encrypted_record[length];
} DTLS\texttt{Ciphertext};
```

- New format for DTLS encrypted traffic
- Can be used like DTLS 1.2 DTLS\texttt{Ciphertext}
- Keyed on version negotiation as expected
Short headers 2: Special DTLSShortCiphertext

```
struct {
    uint16 short_epoch_and_sequence; // 001ESSSS SSSSSSSSS
    opaque encrypted_record[remainder_of_datagram];
} DTLSShortCiphertext;
```

• E == truncated epoch
• S == truncated sequence
• Can only be used
  – With 1-RTT data
  – When you have one record per packet
Reconstructing the epoch/sequence

Sequence reconstruction (same as QUIC):
Use full sequence number closest to seq of the highest successfully deprotected record.

Epoch:
If epoch low-order bits match, just decrypt
If epoch low-order bits match, use the epoch which provides the closest reconstructed sequence number.
Empty Acks

- Sometimes you can’t decrypt part of a flight
  - E.g., you get EE before SH

- In these cases you can’t ACK
  - And rely on the retransmit timeout

- In this case you should send an empty ACK
  - This shortcuts the retransmit
KeyUpdate

- Restored KeyUpdate mechanism
  - Works just like TLS 1.3
  - With ACK, this works properly

- When can you send with the new key?
  - Currently right away
    - What about reordering?
    - ... trial decryption or drop the packet
  - Alternative: can’t send until ACKed
    - Different than with TLS 1.3
    - Arguably less complex (though complexity is on updater)
Remaining Open issues: None!

- WGLC?