DTLS for SCTP

Magnus Westerlund
John Preuß Mattsson
Claudio Porfiri

IETF 122 2025-03-17

IPR Declarations

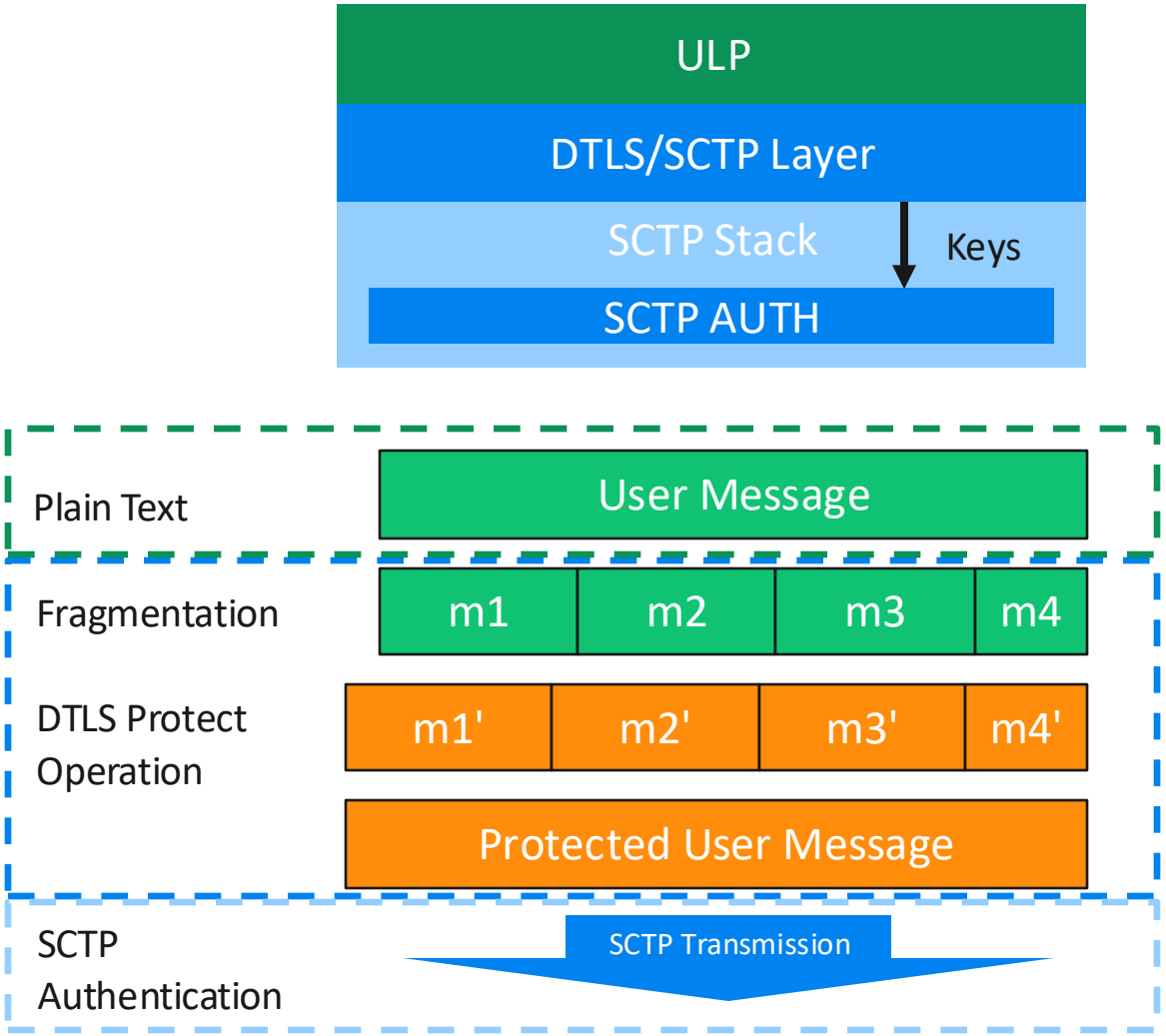
- Option 1: draft-ietf-tsvwg-dtls-over-sctp-bis
  - https://datatracker.ietf.org/ipr/5195/
  - https://datatracker.ietf.org/ipr/6218/
- Option 3 and 4: draft-westerlund-tsvwg-sctp-dtls-chunk-04
  - https://datatracker.ietf.org/ipr/6219/
- Option 3: draft-westerlund-tsvwg-sctp-dtls-handshake-04
  - https://datatracker.ietf.org/ipr/6220/

The missed requirement

- We from Ericsson realized that we did miss an important requirement to discuss in the requirements discussions
- To avoid requiring formal verification
  - We have mentioned that we do not want modified security mechanisms as the track record for TSVWG defined security mechanisms is not that good
  - Formal verification help avoid issues but takes time
  - Time, we don’t see exists for the DTLS for SCTP work
- Think SCTP-AUTH or TSVWG specification of the DTLS record likely to require formal verification to ensure security
- Recertification of implementations?
  - Another aspect is that modified implementations can require certification work
  - Delays delivery and increases cost to deploy

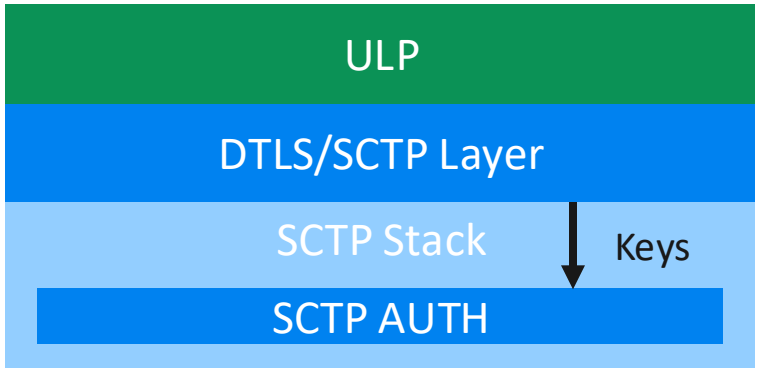
Option 1 (A): DTLS over SCTP

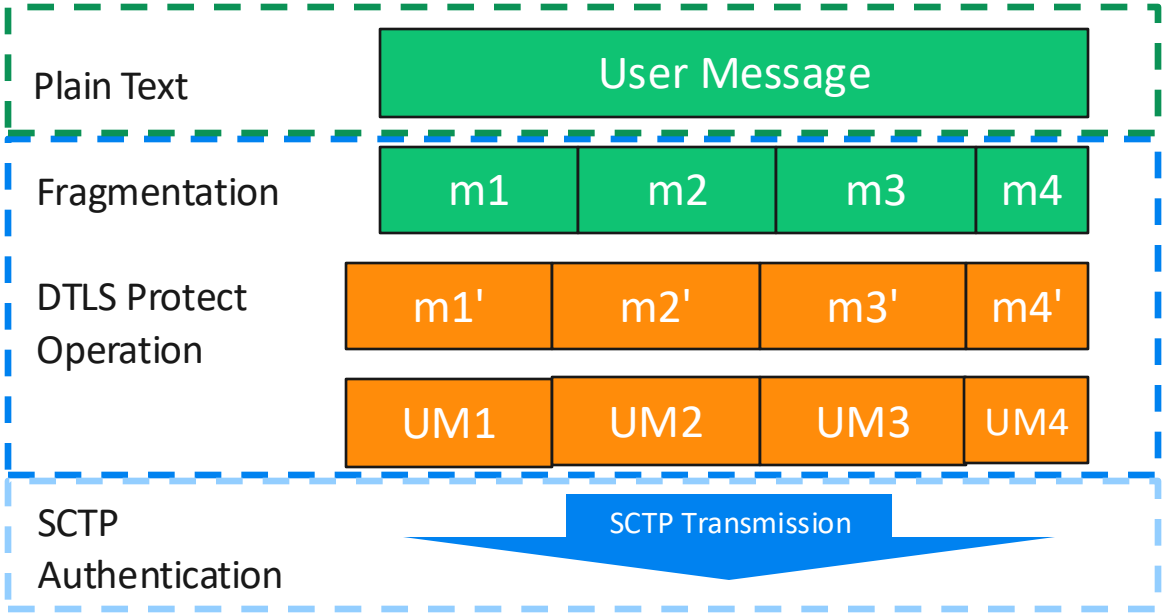
- DTLS over SCTP (SCTP/DTLS)
  - draft-ietf-tsvwg-dtls-over-sctp-bis-08
  - draft-ietf-tsvwg-rfc4895-bis-04
  - An Adaptation layer between Upper Layer Protocol (ULP)
  - Fragments user messages into multiple fragments
  - Each fragment protected by a DTLS record
- Relies on SCTP-AUTH to ensure record order
  - DTLS keys the SCTP-AUTH
- Uses parallel DTLS connection for rekeying
  - Use DTLS Connection IDs for identification



Option 2 (C): DTLS over SCTP

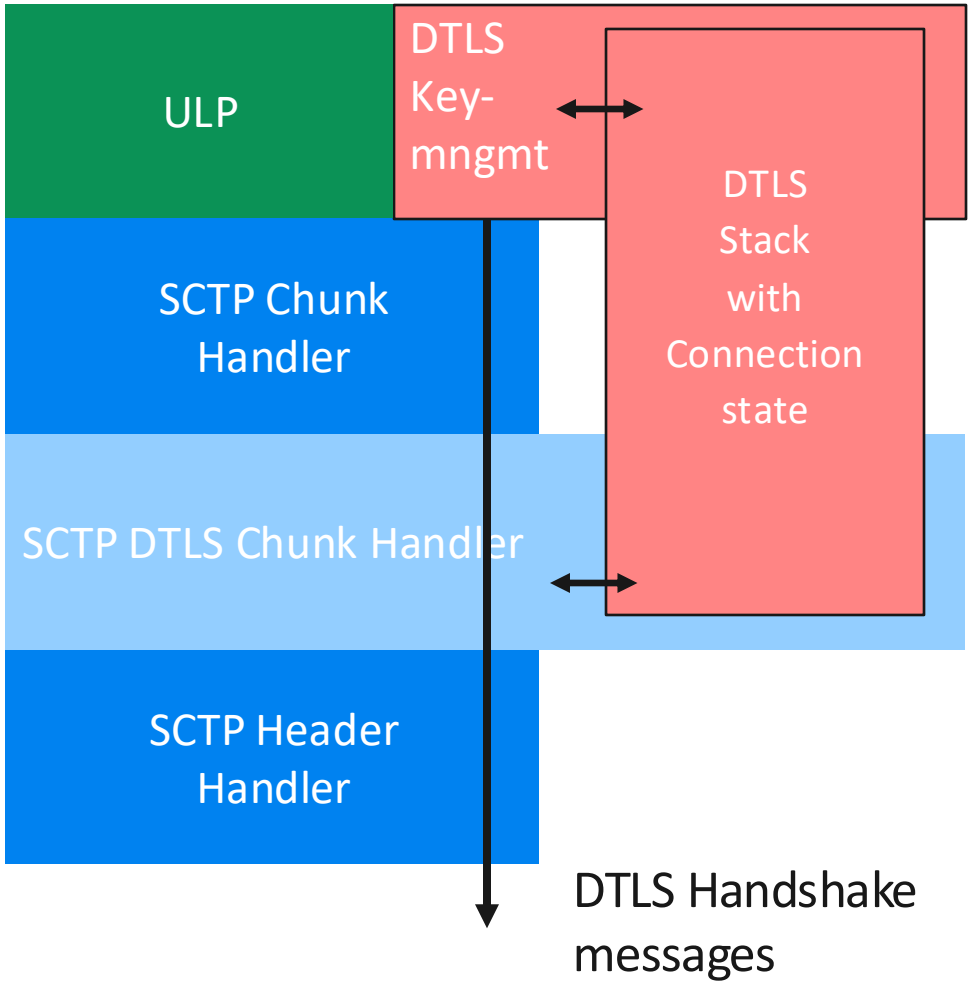
- DTLS over SCTP (SCTP/DTLS)
  - draft-tuexen-tsvwg-rfc6083-bis
    - Rekeying using: https://datatracker.ietf.org/doc/draft-ietf-tls-extended-key-update/
    - Re-authentication using RFC 9261
    - Fragmentation: PPID based fragmentation
  - draft-ietf-tsvwg-rfc4895-bis-04
  - An Adaptation layer between Upper Layer Protocol (ULP)
  - Fragments user messages into multiple fragments
  - Each fragment protected by a DTLS record
- Relies on SCTP-AUTH to ensure record order
  - DTLS keys the SCTP-AUTH
- Uses (D)TLS mechanism for rekeying





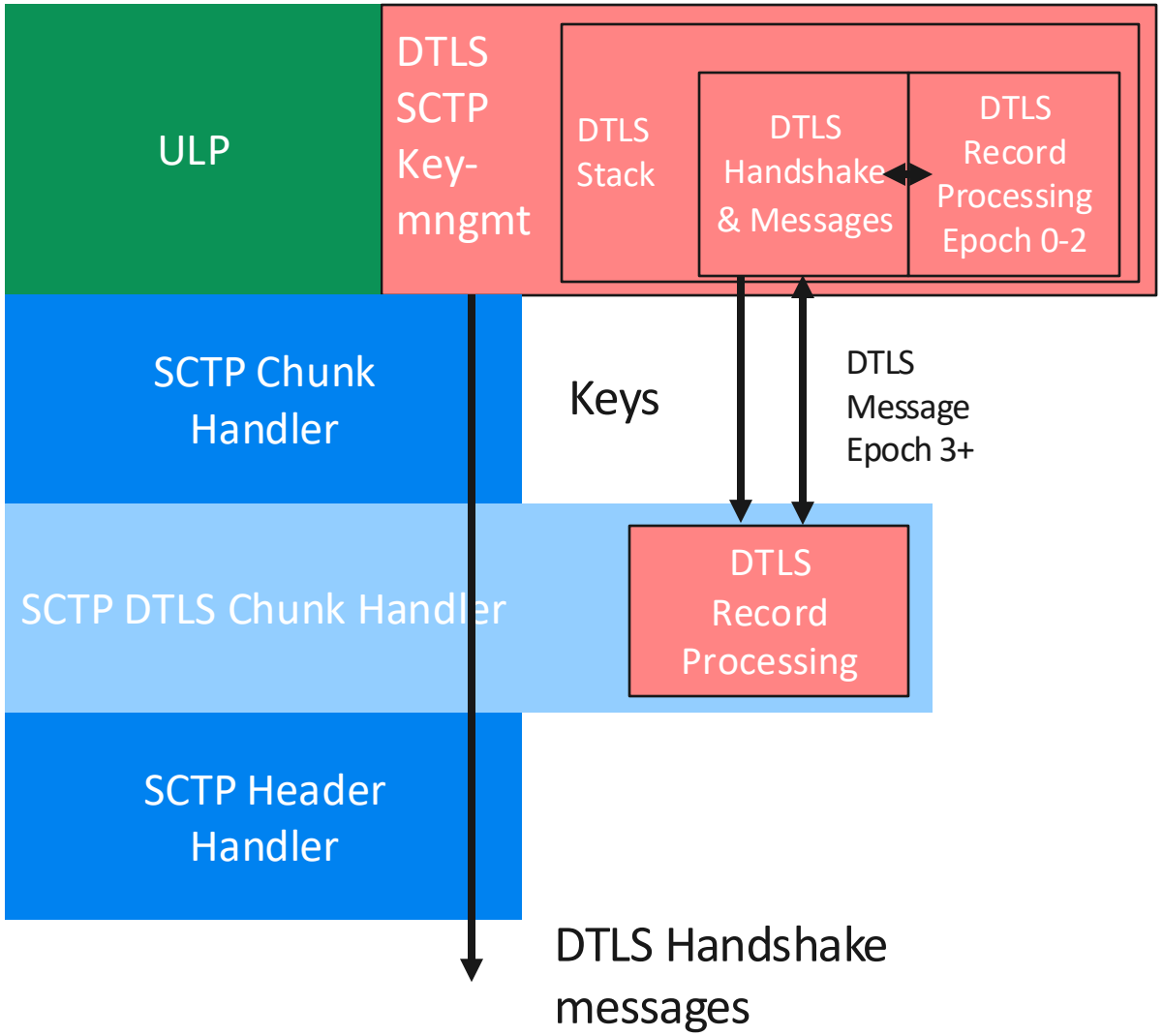
Option 3 (B): DTLS Chunk and Handshake Overview

- DTLS in SCTP
  - draft-westerlund-tsvwg-sctp-dtls-chunk-04
  - draft-westerlund-tsvwg-sctp-dtls-handshake-04
- DTLS Chunk is a new SCTP Chunk
  - Encapsulates all other chunks on a per SCTP packet basis in a DTLS record
- DTLS handshake is run over SCTP as messages
  - As same DTLS state and sequence number space are for DTLS messages and DTLS chunks, the DTLS messages appear to be delayed -> Requires large enough replay window
- User Message Fragmentation done by SCTP's normal message fragmentation mechanism



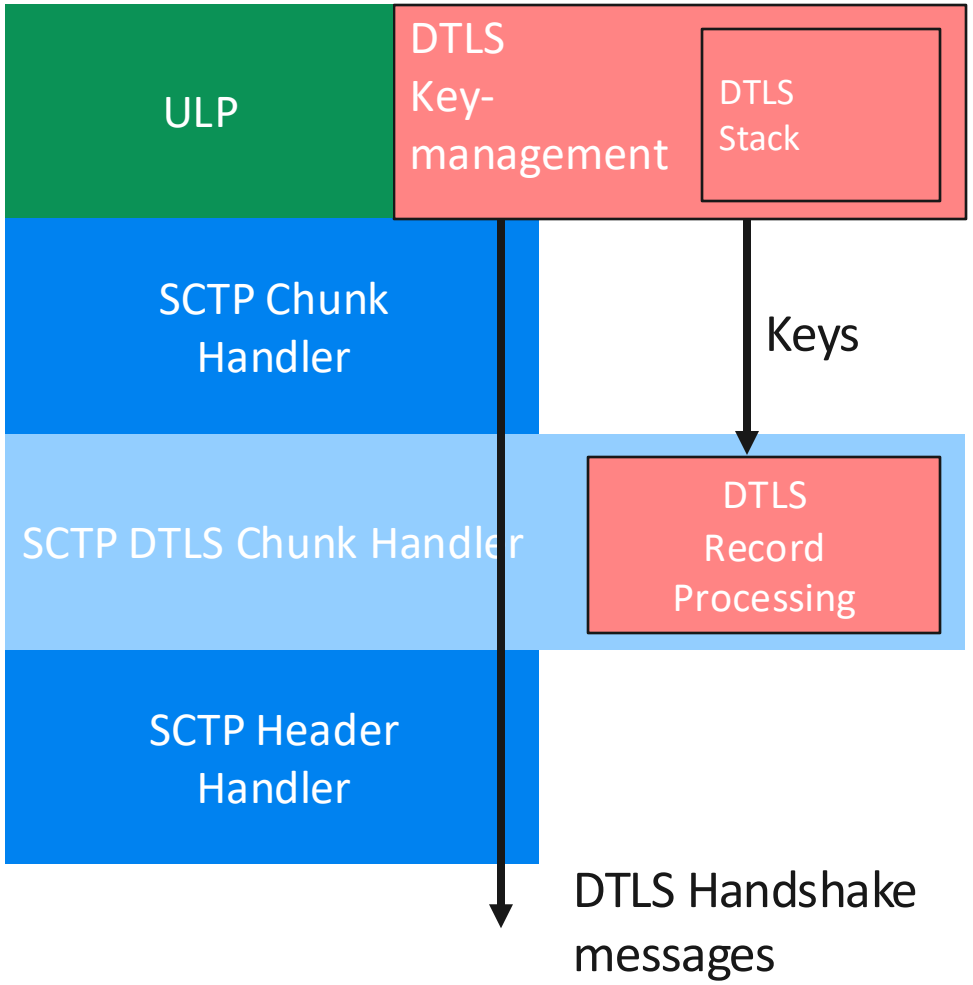
Option 3 (B): Kernel SCTP w/o DTLS in kernel DTLS Chunk and Handshake Overview

- DTLS in SCTP
  - draft-westerlund-tsvwg-sctp-dtls-chunk-04
  - draft-westerlund-tsvwg-sctp-dtls-handshake-04
- Kernel implementation
  - Requires modified DTLS implementation
    - Select which record processing implementation to use
    - But not standard changes
  - For Epoch 3 and forward
    - State in kernel
    - DTLS messages needs to be protected using state in kernel
    - Added abstract API to protect messages to capture this need



Option 4:
DTLS Chunk and Handshake Overview

- DTLS in SCTP
  - draft-westerlund-tsvwg-sctp-dtls-chunk-04
    - Modified to have stand alone DTLS record processing layer in Chunk handler for epoch 3+
    - TSVWG record processing layer description from DTLS 1.3 into TSVWG specification plus interface
      -> Formal verification
    - Maintaining Cipher for the processing layer?
  - Stand alone DTLS session for keys
    - Use TLS key-derivation to create record layer keys
    - Details not agreed on regarding rekeying and re-authentication
- DTLS Chunk is a new SCTP Chunk
  - Encapsulates all other chunks on a per SCTP packet basis in a DTLS record



Unclear how to progress from here?

- What can we do to reach any agreement
  - Who does care about timely specification?
- Can we accept two solutions?
  - One that can be concluded quickly
  - One that are more easily implemented in
    - Kernel space
    - Believed to have less IPR licensing requirements
- Time to send LS to 3GPP?
  - Believe it is time we report back to 3GPP where we are and what we believe on way forward.