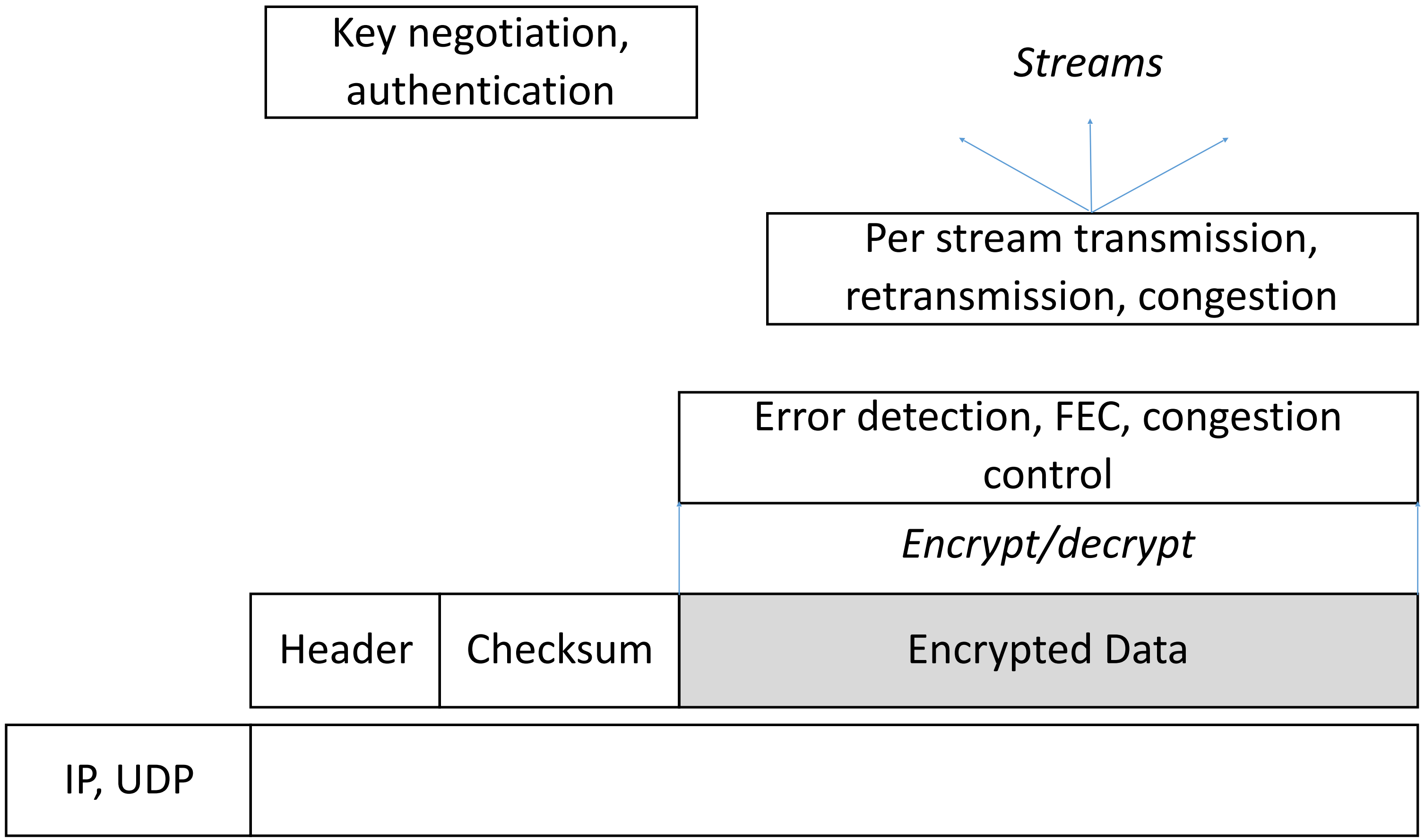
DTLS as a Subtransport

Christian Huitema

Eric Rescorla

Jana Iyengar

Why DTLS?


Key negotiation, authentication
Streams
Per stream transmission, retransmission, congestion
Error detection, FEC, congestion control
Encrypt/decrypt
Header
Checksum
Encrypted Data
IP, UDP

- Encryption essential part of transports
- Implementing own encryption stack is hard
- Design special purpose stack is risky
- DTLS reuses TLS, provides common encryption layer

Gaps: DTLS as efficient sub-transport

- Zero RTT setup
  - Zero RTT setup will be supported in TLS/1.3
- Low overhead
  - 13 bytes of header, out of 1500 bytes UDP packet, maybe too much
  - Could use compression per draft-modadugu-dtls-short-00
- DOS resilience – without TCP 3 ways handshake
  - Resource at server: use DTLS cookies mechanism
  - DOS amplification: require padding of initial packet
- Context-ID
  - Additional 8 byte identifier would allow MP-TCP like functionality
  - Discuss – do we need to multiplex many connections per 5-tuple?

Gaps: DTLS and being middle-box friendly

- Protocol detection
  - Have middle box understand what protocol is being used
  - Some minimal support in DTLS – pattern matching clear text headers
  - But generally goes against the whole point of encryption
- Start-Stop indication
  - Suggested to help resource control at middle box
  - Start is obvious – first packet does it
  - Plausible Stop heuristic, monitor “Alert” content type in clear text header
- Accepting indications from the network
  - For example, “congestion detected” or “PMTU supported”
  - Goes against the grain of DTLS – not encrypted, not secure
  - Heuristics are possible, but not obvious
  - End-to-end evaluation of MTU, congestion seems more plausible

To do

- Build actual prototype
- Feedback from SPUD BOF