QUIC + TLS

draft-thomson-quic-tls-01
IETF 97

Rationale

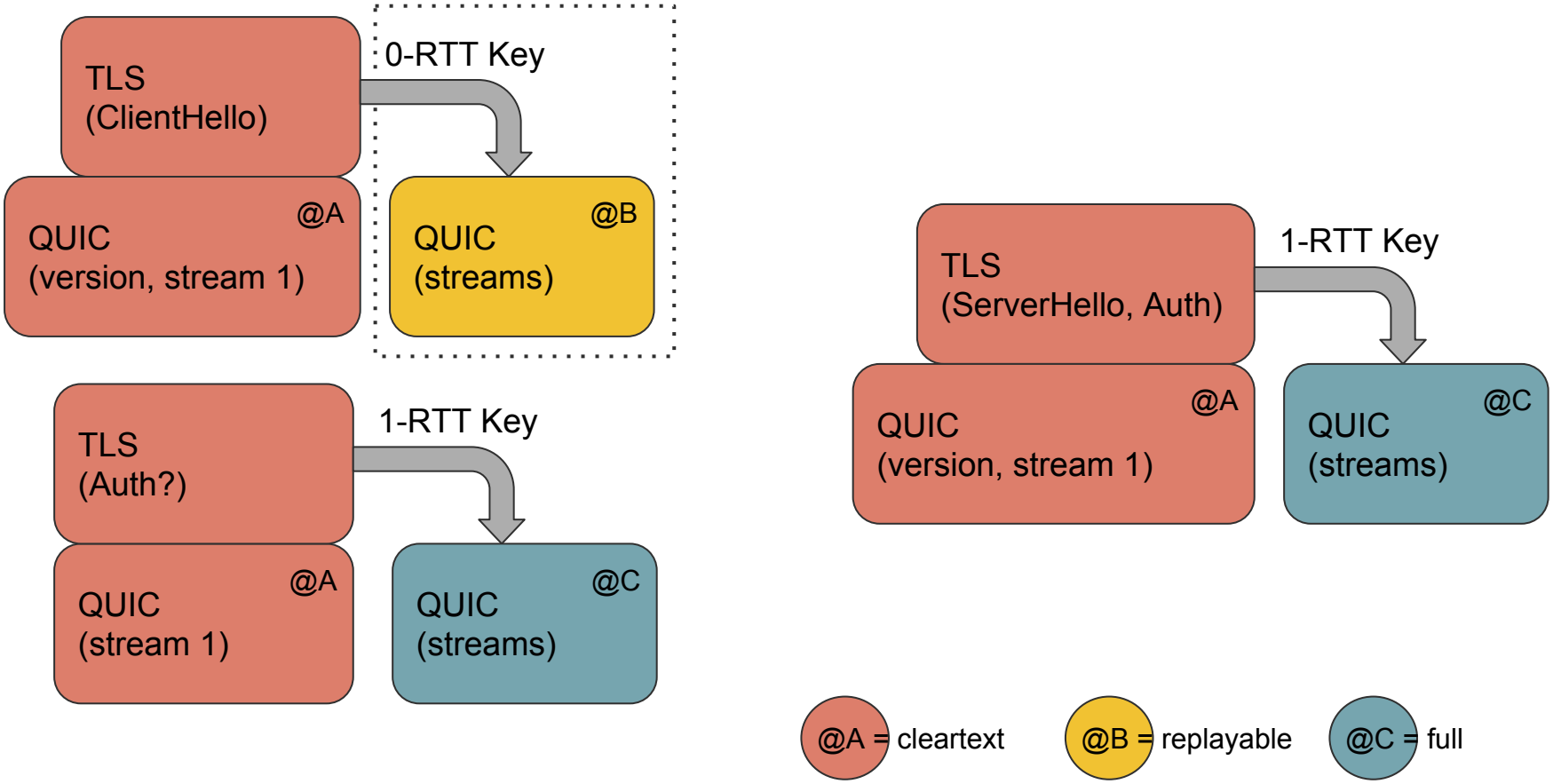
QUIC does reliable, in order delivery

TLS needs reliable, in order delivery

TLS does key exchange, w/ 0-RTT

QUIC needs key exchange, w/0-RTT

Handshake


TLS
(ClientHello)
0-RTT Key
QUIC
(version, stream 1)
@A
QUIC
(streams)
@B
TLS
(Auth?)
1-RTT Key
QUIC
(stream 1)
@A
QUIC
(streams)
@C
TLS
(ServerHello, Auth)
1-RTT Key
QUIC
(version, stream 1)
@A
QUIC
(streams)
@C
@A = cleartext
@B = replayable
@C = full

Encryption

Full TLS on stream 1

- That includes all records…
- … and TLS encryption (esp. handshake)
- Double-encryption limited to a few messages (NewSessionTicket basically)
- **Q: Should stream 1 always be in the clear?**

TLS exports the keys that QUIC uses, QUIC manages packet protection

- Packet protection modelled on DTLS

KEY_PHASE

KEY_PHASE avoids trial decryption (as used in the existing code)

In 1-RTT, all packets up to Finished message are sent in the clear

- Cleartext packets have KEY_PHASE=0

After writing the Finished message is sent

- Disable cleartext for everything **except retransmission of stream 1 data**
- Change to writing with 1-RTT keys and mark packets with KEY_PHASE=1

After reading KEY_PHASE=1, change to reading with 1-RTT keys

KEY_PHASE 0-RTT
(as previously proposed)

Client sends QUIC handshake and TLS ClientHello (KEY_PHASE=0)

Client changes to 0-RTT keys after sending ClientHello (KEY_PHASE=1)

Client's second flight of TLS handshake is sent in the clear (KEY_PHASE=0)

Once TLS handshake completes, move to 1-RTT keys (KEY_PHASE=1)

Server is easy (the Server doesn't use 0-RTT keys)

0-RTT Problem

Situation: The client's second flight is lost

Packets encrypted with different keys (0-RTT and 1-RTT) arrive at the server

These packets are all marked KEY_PHASE=1

The server needs to distinguish between three keys (cleartext, 0-RTT, 1-RTT)

A. Trial decryption just this once (try with both keys)
B. Steal another bit
C. Rearrange the QUIC header somehow
D. Overload the version bit (define KEY_PHASE=1+VERSION=1 as 0-RTT)
E. Encrypt the client's second flight with the 1-RTT keys
F. Something else even more clever

Proposal

ClientHello - KEY_PHASE=0 VERSION=1

Early data - KEY_PHASE=1 VERSION=1

Client Finished - KEY_PHASE=0 VERSION=1

Application data - KEY_PHASE=1 VERSION=0

Cost: more overhead for 0-RTT

Note that you could encrypt Client Finished and use KEY_PHASE=0 VERSION=0

You could also use VERSION=0 for early data (i.e. version == “encrypted” bit)

KeyUpdate

TLS 1.3 defines a KeyUpdate message for refreshing keys, however

QUIC keys are independent of those in TLS

KeyUpdate design assumes reliable, in-order delivery

Proposal:

1. Forbid use of TLS KeyUpdate
2. Use KEY_PHASE to indicate refresh of write keys
3. Endpoints must update both keys so that number of refreshes is the same
   i.e., make KEY_PHASE the same
4. Forbid a second update until peer has refreshed in response