Key Exchange Customization for TIPTOP

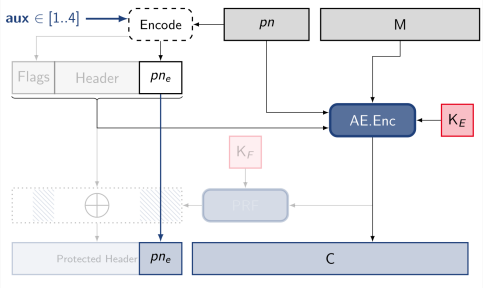
Implementing MLS inside QUIC

Benjamin Dowling benjamin.dowling@kcl.ac.uk
Britta Hale britta.hale@nps.edu
Xisen Tian xisen.tian1@nps.edu
Bhagya Wimalasiri b.m.wimalasiri@sheffield.ac.uk

March 20, 2025

QUIC Handshake and Record Protocols

On a high-level QUIC operates as two sub-protocols: **Handshake** and **Record** Protocols



Record Protocol slide courtesy of Felix Günther

Space & Interplanetary Communication

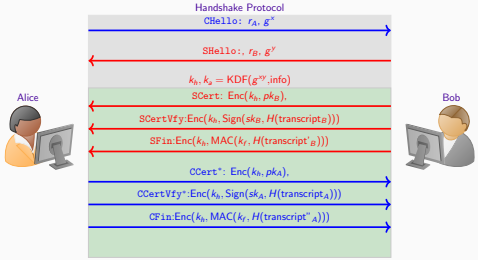
The terrestrial internet as we know it operates under certain assumptions.

Terrestrial Communication	**Interplanetary Communication**
Continuous availability of links and devices	Discontinuous availability
Low latency and fast RTT, e.g., RTT between two points is approximately 100 to 300 ms	Latency
Low packet loss rate	Packet loss, attenuation, jamming potential
Bi-directional data links	Data links can be simplex
Ubiquitous client-server model	Handshakes are unreliable/costly

Session establishment handshakes can be unreliable
packet loss incurs further delays in key establishment

Proposed Solution: QUIC

QUIC at its core is a secure channel establishment protocol based on TLS:

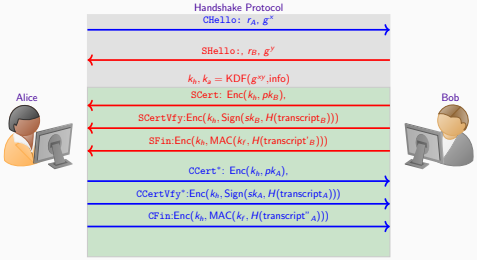


- **Forward Secrecy**: Compromise of long-term keys don't impact previous communciation security.
- **Provides Key Update**: Alice and Bob can ratchet forward their secrets $\mathtt{secret}_{n+1} = \text{HKDF-Expand}(\mathtt{secret}_n, \mathtt{quic\ ku}, \emptyset, \text{H\_len})$.

But, TLS **isn't** a good choice for the asynchronous setting.

Proposed Solution: QUIC

QUIC's handshake requires synchronous communication:



- **No Post-Compromise Security**: Future Confidentiality lost once keys compromised.
- **High Risk in Low Latency**: 0-RTT mode has known issues (see RFC 8446)
- **Assumes Synchronous Communication**: Alice and Bob both must be interactive to perform handshake.

But, once we establish keys, we could use Key Update or Extended Key Update, right?

QUIC Key Update

QUIC's key update mechanism does not refresh randomness

Figure 9 shows a key update process, where the initial set of keys used (identified with @M) are replaced by updated keys (identified with @N). The value of the Key Phase bit is indicated in brackets [].

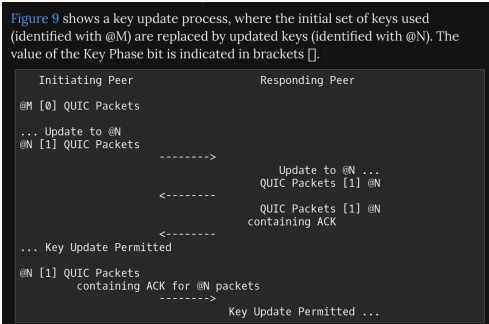


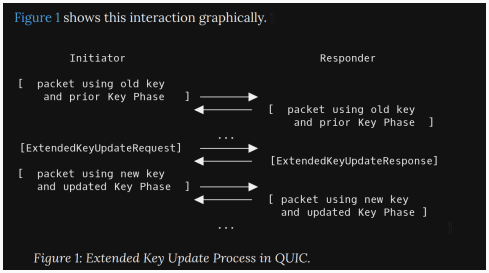
Figure 9: Key Update

- **Provides Key Update**: Alice and Bob can ratchet forward their secrets $\texttt{secret}_{n+1} = \text{HKDF-Expand}(\texttt{secret}_n, \texttt{quic ku}, \emptyset, \text{H\_len})$.
- **Forward Secrecy**: Compromise of previous secrets don't impact previous communication security.
- **No Post-Compromise Security**: Future Confidentiality lost once keys compromised.

But QUIC has extended key update, right?

QUIC (Extended) Key Update

QUIC's extended key update mechanisms uses Diffie-Hellman to inject new entropy:

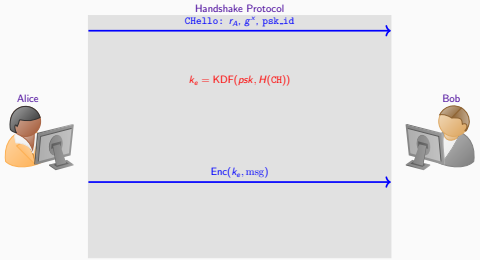


- **Post-Compromise Security in Question**: Not sure it has been formally analysed.
- **Assumes Synchronous Communication**: Alice and Bob both need to be online to perform handshake.
- **NEVER analyzed as a CKA**: "spreading out" the handshake/update is outside scope for prior analyses, and key compromise mid-handshake not modeled.

But QUIC has 0RTT, right?

QUIC 0RTT

One potential solution proposal that has been floated to support **asynchronous communication** is 0RTT.



However:

- *TLS does not provide inherent replay protection for 0-RTT data.*
  - very much leaves things to application/implementor...
- **Loses Forward Secrecy**: Once the PSK is leaked, all messages using PSK are known.

QUIC Is Going to Get Worse

- QUIC handshake bandwidth costs are about to get a lot worse.
- Making QUIC quantum resistant will incur **considerable session establishment costs** from given the size of PQ KEMs and signatures.

Which all led us to the question...

...Is QUIC the right protocol here?

Use of timing profiles, support for asynchronicity, recovery from packet loss, etc. implies an demand signal for **Continuous Key Agreement (CKA)**

- BUT TLS (and therefore QUIC) are not CKAs – they are session-based protocols.
- The analysis of TLS focuses on the **per-session guarantees.** Their composition is also analysed, but these guarantees are broadly session-oriented and do not model security in the "spread out" handshake sense of a CKA.
- Trying to change QUIC into a CKA requires a new analysis under a different model (which we already know it would fail) but is also re-inventing the wheel...

...Why not use an existing standardized CKA? **RFC9420, Messaging Layer Security**.

(Bonus: this only changes the QUIC handshake**
the rest of the QUIC protocol can be used as-is)

Why MLS?

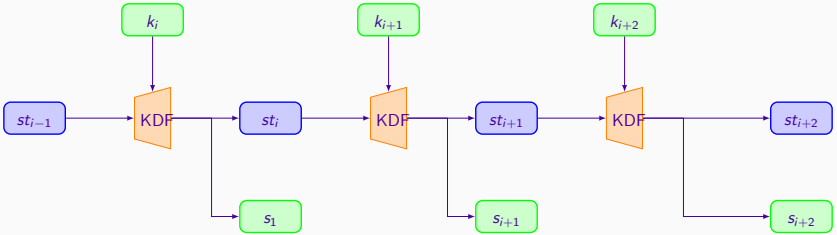
Goals – Setting

- **secure continuous key agreement protocol**
- **Asynchronous**, key updates can be unidirectional
- **Sessions are long-lasting**, can evolve updates over even years
- Low bandwidth & complexity
- (Optional) Can handle groups of devices and dynamic group membership

Goals – Security

- E2E Authenticity and Confidentiality
- Adversarial Model
  - MiTM: Network, Delivery Service compromised
  - Corruption: Can heal from state compromise
  - Adaptive and Active Adversary
- **Forward and Post-Compromise Security**

MLS: Epochs



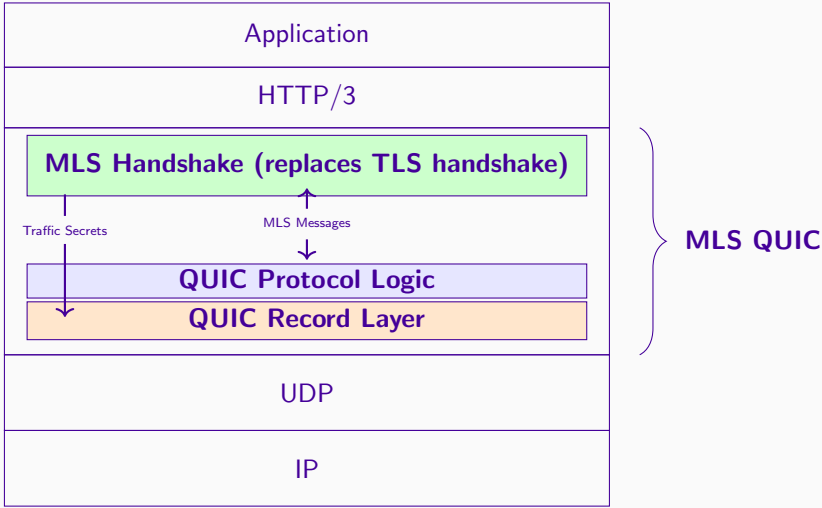
- Keys k found at the root of a Ratchet Tree are called "commit secrets" and are used to generate keys for encrypting messages.
- Members can "Update" the tree, generate a new commit secret k_{i+1}, which advances the group to the next epoch $i + 1$
- Commit secrets k_i are combined with secrets generated from the previous epoch to achieve post-compromise security

(The key schedule has more too it than this... this is an overly simplistic view but gives a sense of CKAs)

Proposal: QUIC-MLS

- Take advantage of work done to fit QUIC
- Replace **QUIC Handshake** with **MLS Updates**
- Achieve PCS and Asynchronous Updates
- be resilient against delays from session setup

MLS Integration within QUIC



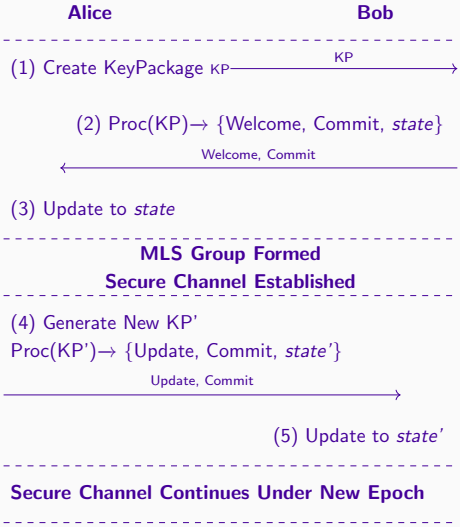
PoC and Benchmarking

PoC & Benchmarking Specs

- Testbed: 11th Gen Intel(R) Core(TM) i7-11370H @3.30GHz with 16GB onboard RAM
- PoC developed using quic-go and cisco/go-mls libraries

Security Function	QUIC-TLS	QUIC-MLS
Handshake/Join Group	1.7 ms	2 ms
PCS Key Update	N/A	1.7 ms

Table 1: *Performance Comparison between QUIC-TLS and QUIC-MLS Cryptographic Operations*



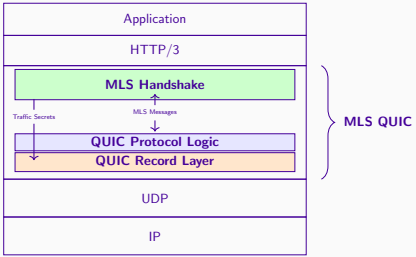
Summary

Core Idea:

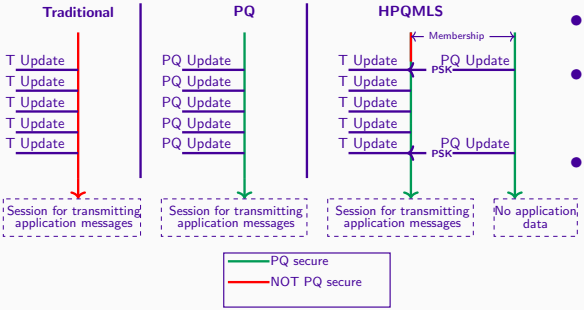
- Replace QUIC handshake with MLS handshake.
- QUIC Record Layer remains the same.
- QUIC adoption + MLS security.
- **Is the WG interested in this approach?**

Next Steps:

- If WG interest in this approach, WG chairs will coordinate with QUIC and we will take this to QUIC WG for draft and adoption/work.
  - QUIC-MLS extension.
- Return result to TIPTOP for adoption.



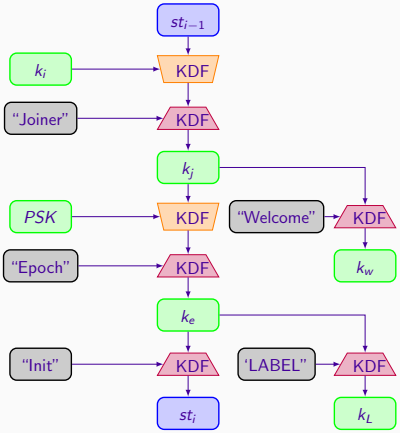
One Step Further? Post Quantum MLS



- PQ can be costly
- Hybrid strikes a balance between security & efficiency
- Potential amortized design
  - Use two sessions – T and PQ
  - combine via exported PSKs.
  - **Flexibility**: PQ updates are tunable

MLS Key Schedule

However, MLS isn't quite this simple - we combine the updated i-th commit secret k_i with the previous "secret state" st_{i-1} to generate keying material for various purposes.



k_L	Label
sender_data_secret	"sender data"
encryption_secret	"encryption"
exporter_secret	"exporter"
authentication_secret	"authentication"
external_secret	"external"
confirmation_key	"confirm"
membership_key	"membership"
resumption_secret	"resumption"