



Instantiating PQ Cryptographic Protocols

Status, Challenges and Solutions for the PQ Transition

SPIQE

Benjamin Dowling

2026-07-10

benjamin.dowling@kcl.ac.uk

About Me - Dr. Benjamin Dowling (Ben)



- **Senior Lecturer in Cryptography** (Cybersecurity group)
- Research focus: **Applied Cryptography**
 - Specialize in real-world key exchange and communication protocols,
 - i.e. **TLS, Signal, WhatsApp, SSH, Matrix, Noise, LINE**, etc.
 - **Post-Quantum**: Introduced PQNoise, Muckle, analysed PQ-SSH.
- Previously at Sheffield, ETH Zurich (Applied Cryptography), Royal Holloway (ISG).

Background

Post-Quantum Signal

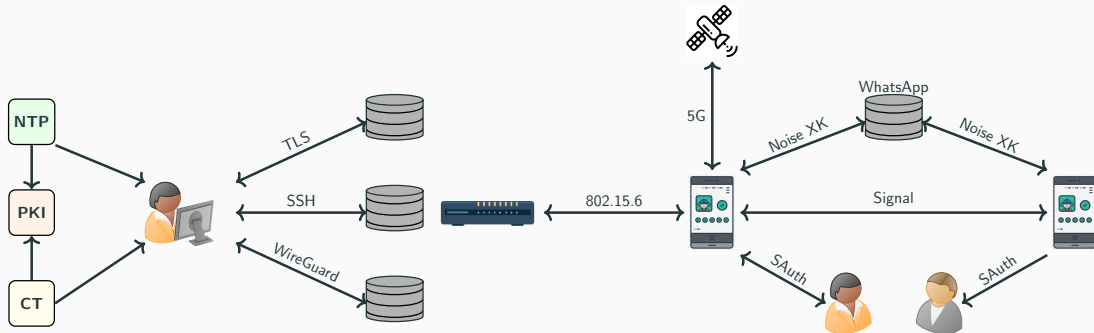
Post-Quantum TLS

Post-Quantum QKD

Conclusions

Securing Communication by Cryptography

We use cryptography and **cryptographic protocols** to secure communication online.



Quantum computers are coming: How do we transition to a **post-quantum** future?

Standardisation of Post-Quantum Cryptography

Post-Quantum Cryptography is **maturing**: Standardisation processes are now **finalising**.



FIPS 203

[Module-Lattice-Based
Key-Encapsulation Mechanism Standard](#)
(ML-KEM)



FIPS 204

[Module-Lattice-Based
Digital Signature Standard](#)
(ML-DSA)



FIPS 205

[Stateless Hash-Based
Digital Signature Standard](#)
(SLH-DSA)

BSI – Technical Guideline

Designation:	Cryptographic Mechanisms: Recommendations and Key Lengths
Abbreviation:	BSI TR-02102-1
Version:	2026-01
As of:	January 23, 2026

We have standardised **post-quantum key establishment** and **post-quantum digital signature schemes**. So, all we need to do is utilise them? Lets look at some challenges for this approach.

Post-Quantum Candidates

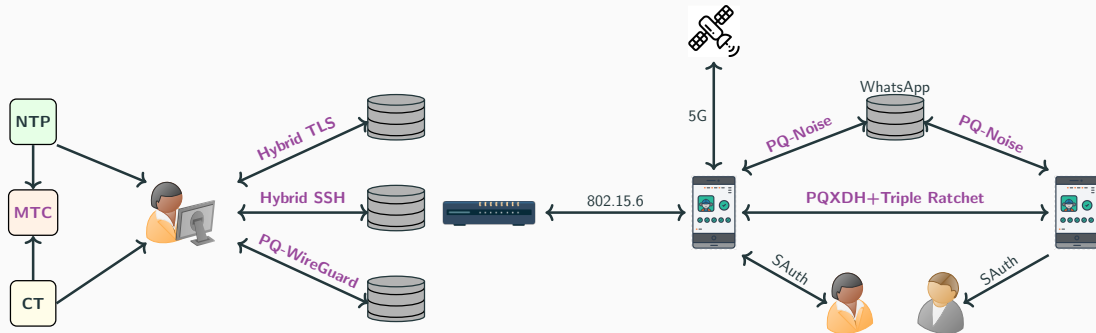
Name	Type	ctxt (B)	sk (B)	pk (B)
ECDH	KEM	32	32	32
ML-KEM	KEM	768	1,632	800
FrodoKEM	KEM	9,752	19,888	9,616
HQC	KEM	4,433	2,321	2,241
ECDSA	DS	32	32	32
ML-DSA	DS	3,293	32	1,952
Falcon	DS	666	2,000	897
SLH-DSA	DS	7,856	64	32

Challenges with PQ transition:

- **BANDWIDTH:** bandwidth **more expensive** for PQC, even for “nice” replacements.
- **CONF vs AUTH:** Need PQ conf. (KEMs) ASAP for **HNDL**, PQ auth. (DS) comes later.
- **NIKE vs KEM:** KEMs not a straightforward replacement for Diffie-Hellman!
- **PQ HESITANCY:** Concerns about PQ maturity $\implies$ **hybrid** approaches.

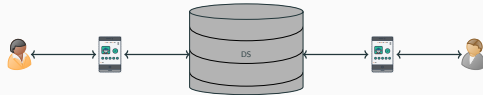
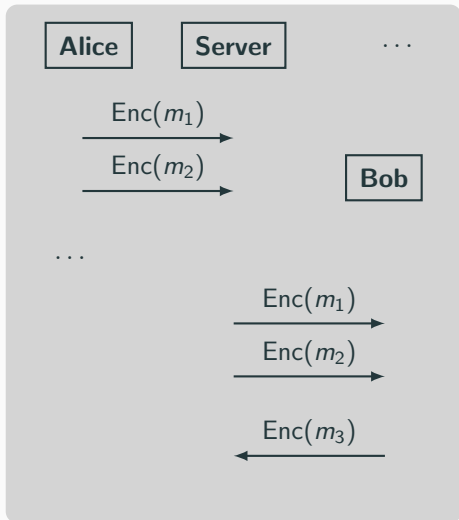
Cryptographic Ecosystems

This transition has started for many protocols, but not **consistently**: many sit at different stages.



The post-quantum transition is also not **uniform**. Lets look at some examples.

Signal - A Two-Party Asynchronous E2EE Messaging



Two-party Asynchronous E2EE messaging protocol:

- **E2EE** - Only Alice and Bob able to read messages
- **Asynchronous** - Parties not required to be online at the same time
- Server now needed to handle message delivery
- **Key Establishment** needs to be async., i.e. **non-interactive**: DH?

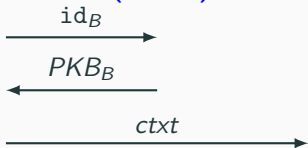
Signal Messaging Protocol Overview



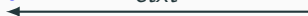
Reg Phase



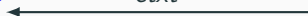
Init Phase (X3DH)



Asym Phase



Sym Phase



Consists of three (major) phases:

- **Registration Phase**: Parties upload public keys to Server
- **Initialisation Phase**: Parties retrieve public key bundles to establish secrets
- **Double Ratchet Phase**: Parties continually update and exchange public keys to establish fresh secrets (**Asym**). Parties then derive keys from these secrets to send messages (**Sym**).

Signal Registration Phase

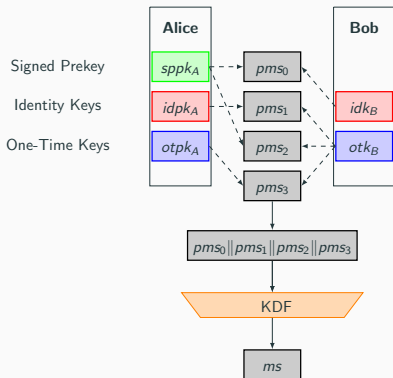
Alice wants people to be able to message her. To do so, she generates a **PreKey Bundle**:

- id_A - long-term identifier (Phone number/Username)
- $(idk_A, idpk_A = g^{idk_A}) \xleftarrow{\$}$ DHGen - long-term public-key pair (Identity key)
- $(spk_A, sppk_A = g^{spk_A}) \xleftarrow{\$}$ DHGen - med-term public-key pair (Signed Prekey)
- $(otk_A, otpk_A = g^{otk_A}) \xleftarrow{\$}$ DHGen - ephemeral public-key pair (One-time Key)
- $\sigma_A \xleftarrow{\$}$ SIG.Sign $_{idk_A}(sppk_A)$ - signature

$$\mathbf{PKB}_A = \{id_A, idpk_A, sppk_A, otpk_A, \sigma_A\}$$

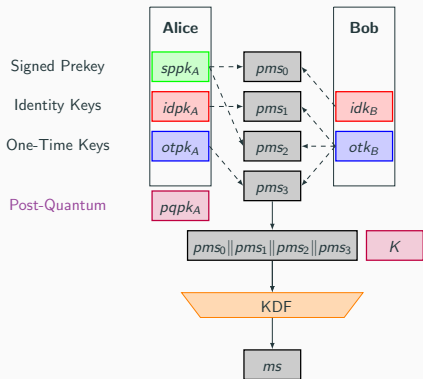
Alice sends the **PreKeyBundle** to the Signal Server. Note: Based on classic DH, classic signatures. No PKI, keys are checked **out-of-band**.

Signal Initialisation Phase (X3DH KEX)



- Bob wants to communicate with Alice.
- Bob gets PKB_A from Server and does eXtended triple (3) Diffie-Hellman to produce a symmetric master secret:
- Bob generates their own one-time DH $(otk_B, g^{otk_B} = otpk_B) \xleftarrow{\$} \text{DHGen}$
- Bob combines PKB_A and idk_B, otk_B using X3DH to compute “master secret” (later used to derive root key rk)
- Heavily based on Diffie-Hellman.

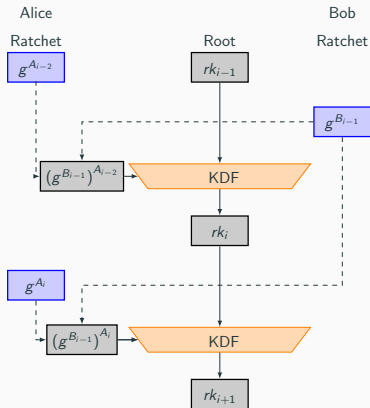
Signal Initialisation Phase (PQXDH)



- **Post-quantum X3DH = PQXDH:**
- X3DH + **PQ-KEM**
- Alice additionally includes **KEM** in her **PKB_A**
- Bob encapsulates under pk_A , computes K
- $ms = \text{KDF}(pms_0 || pms_1 || pms_2 || pms_3 || K)$
- Thus ms achieves post-quantum security.
- Protects against HNDL attacks
 - Does not achieve equiv. PQ security as Signal: no “MEX” security.
- Does not protect against active quantum attackers:
- → **classical signature still used to sign pk_A**

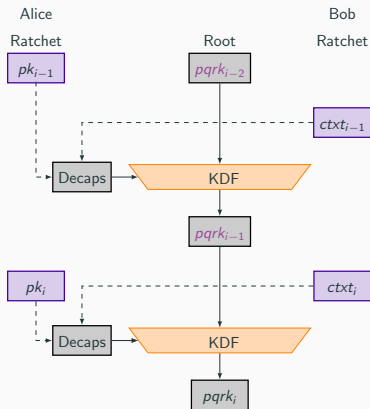
Signal's Double Ratchet Protocol: Exchanging Diffie-Hellman Keys

Once both parties derive root key rk (from master secret ms), Signal wants to **refresh** with new entropy to protect against **state-compromising adversaries**.



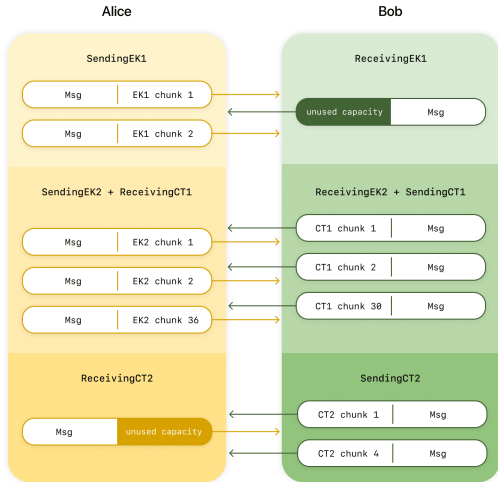
- **Core Idea:** Alice and Bob continually exchange **DH keys**: $A \rightarrow B$, then $B \rightarrow A$, repeat...
- The output is combined with their existing **root secret** to update.
- Resilient against **state compromise**:
 - If attacker learns state rk_{i-1} , once full DH exchange complete, security refreshed and rk_i unknown to attacker!
 - KDF prevents attackers learning state rk_i once "updated" to rk_{i+1}
- However, relies on **DH**, which isn't **post-quantum**.

Signal's SPQR: Sparse Post-Quantum Ratchet



- Add **PQ Ratchet alongside** Double Ratchet
- **PQ Ratchet** based on **PQ-KEMs**, not DH.
- Each update both Alice and Bob must send $pk/ctxt$ respectively.
- **Bandwidth:** **PQ expensive** (large pk , $ctxt$), can't send with **every** message like DR.
- **Unreliable Channel:** messages may drop, re-order.

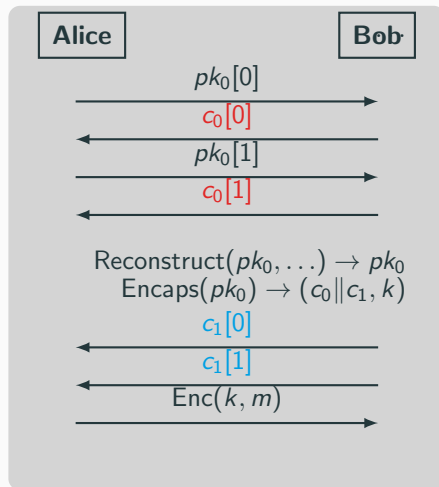
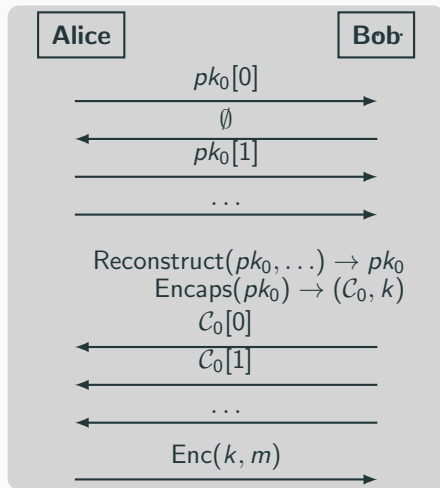
Signal SPQR¹: Sparse Post-Quantum Ratchet



- **Bandwidth:** **Amortise the cost!**
- Split KEM pks and ctexts into “chunks”
- Each msg sends fresh chunk
 - PQ ratchets **periodically**
 - Not every message exchange!
- Recover pk/ctxt after sufficient chunks.
- **Unreliable Channel:** Recover from losing “chunks”? **Erasure Codes**

¹Image Source: Signal (<https://signal.org/blog/spqr/>)

Improving Ratcheting Guarantees



Split KEM into **offline** c_0 (independent of pk) and **online** c_1 (requires pk) (and) $|c_1| < |c_0|$.
Ciphertext c_0 sent earlier, shortening ratchet period.

Lessons from Signal's PQ Transition?

Classical (ECDH)
Compact: ~32 bytes



Post-Quantum (ML-KEM)
Large: ~1–1.5 KB

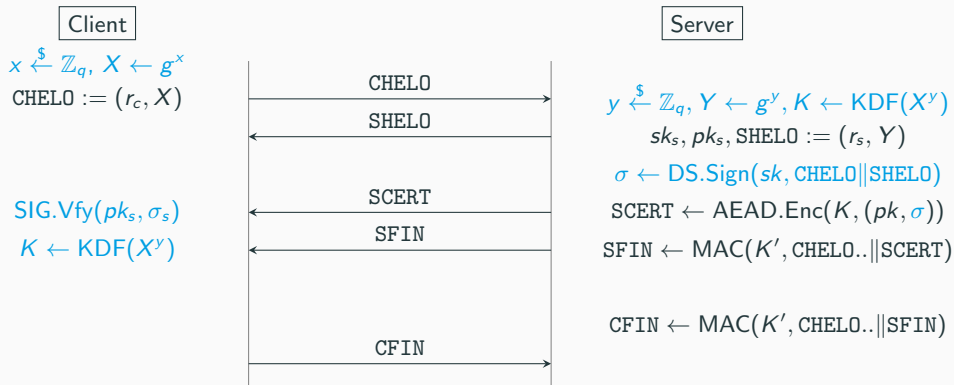


- PQ = expensive, due to **bandwidth**.
- For long-lived sessions, send **chunks** of PQ to establish PQ periodically.
- Erasure codes used for unreliable channels.
- Willing to bend standardisation, can go further.
- Security = fragile: **add, don't replace**.

TL;DR: Post-quantum significantly increases communication cost, motivating **sparse** redesigns.

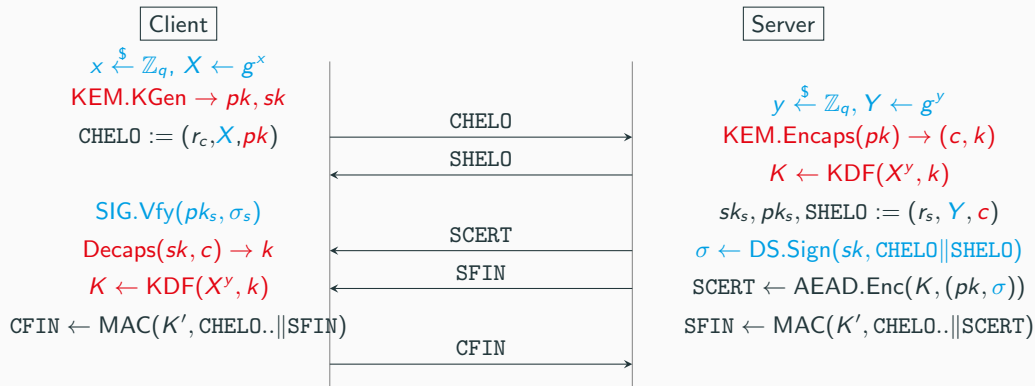
TLS 1.3 Overview

Secure Channel Establishment 1.5RTT protocol achieving confidentiality (encryption), integrity (authentication), forward secrecy (ephemeral key exchange). Structurally, handshake is a **Signed (ephemeral) Diffie-Hellman** at its core.



What is the current approach for **post-quantum TLS**?

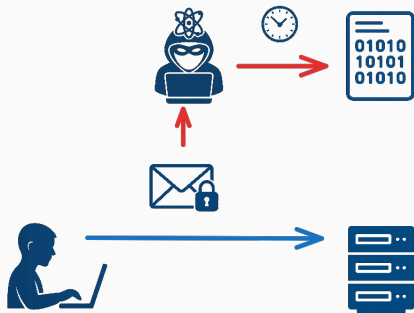
Post-Quantum TLS 1.3



So far, **PQ-TLS**'s approach similar to Signal **PQXDH**: HNDL protections via **PQ-KEM addition**.

Authentication - Harvest Now, Decrypt Later

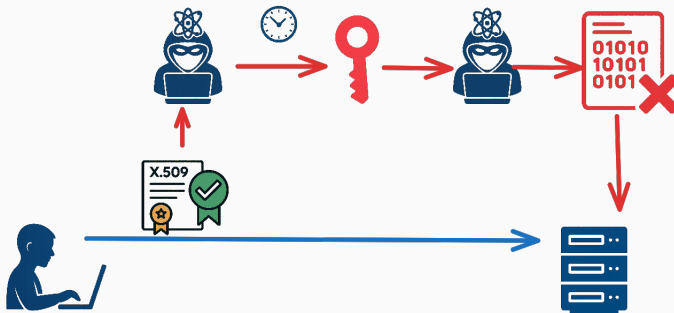
Nearly all approaches we've covered focused on **Harvest Now Decrypt Later Attacks**, i.e. have focused on establishing post-quantum keys, **ignored public-key-based authentication**.



What's the impact of a quantum-equipped attacker against this current approach?

Authentication - Forgery Attacks

Obviously, attacker uses their quantum abilities to recover the classical signature key in a TLS handshake, and then **forges messages** as if they were that party.

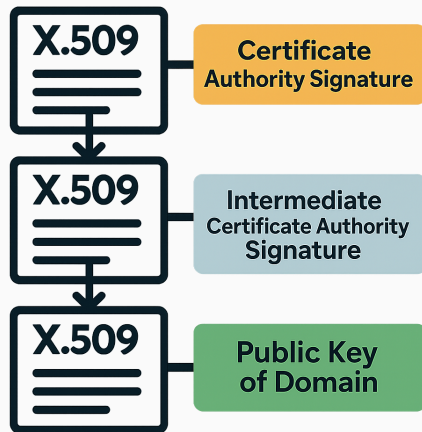


We have post-quantum signatures, so **drop-in replacement possible**. What is the impact of post-quantum on authentication?

Signatures in TLS via Digital Certs

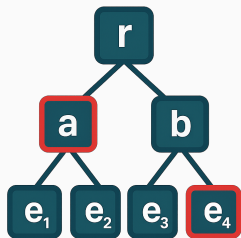


- TLS server signs handshake transcript.
- To authenticate server verification key, server sends a digital cert.
- Digital certificate contains digital signature **chain**.



Thus, need to ensure the whole chain is **post-quantum**. $2 \times pk$, $3 \times \sigma$: **DH** - 256B; **ML-DSA** 18.5kB. **How can we reduce these costs?** New approach: **Merkle Tree Certificates**.

Crypto Background: Merkle Trees

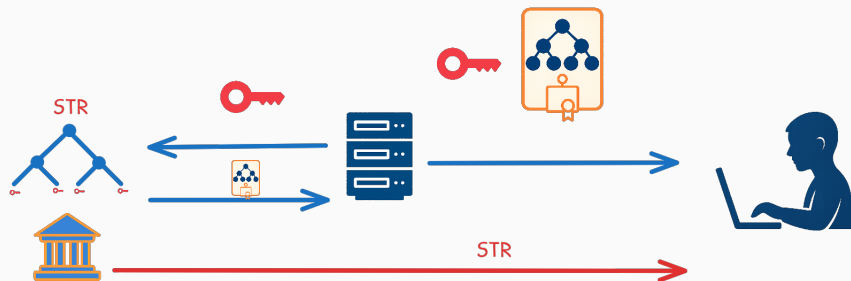


- Merkle Trees are a hash-based binary tree structure
- **Logging scheme**: Prove an entry e is contained in **log**
- Each entry e hashed into a leaf node
- Create a left-balanced binary tree from leaves
- Each node hashes the two descendents

MTs allow efficient representation of $\log r$, and **inclusion proofs**: for some r , reveal “alt path” to the root: $(e_3, H(e_4), a = H(H(e_1)||H(e_2)))$. To **verify**, re-compute hashes, compare with r .

Relevance to Certificates: Can now sign root r , and (with inclusion proof) **authenticate verification keys**.

Merkle Tree Certificates

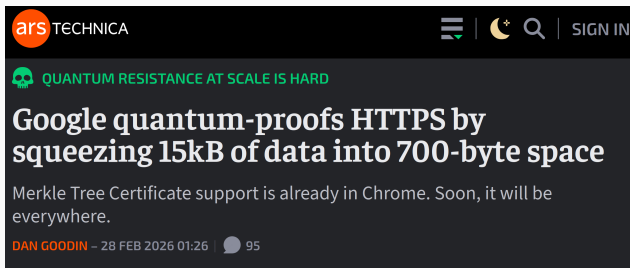


- Certificate Validation = verifying Inclusion Proof in CA Log.
- CAs now batch-sign certificates by signing the Merkle Tree Root.
- Servers receive **Certificate**: inclusion proof, signed tree root.
 - Proof of inclusion = $\log(N)$ hashes, ($< 1\text{kB}$), signed tree root sent out-of-band
 - **Bandwidth**: Removing cert chain $\implies |MTC| < |X509|$
- **Verify**: Reduces from $3x\sigma$, $2xpk$ to $1x\sigma$, $1xpk$, $1x\text{Inclusion Proof}$.

PLANTS and Merkle Tree Certificates

- This approach has its own IETF working group: **PLANTS** (PKI, Logs and Tree Signatures)
- Supported by **Cloudflare and Google**:

Today we're announcing our intent to deploy MTCs on an experimental basis in collaboration with Chrome Security. ²

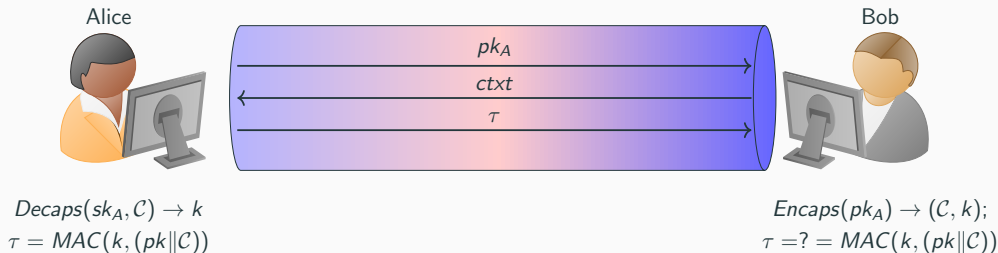


²Valenta et. al. (<https://blog.cloudflare.com/bootstrap-mtc/>) Cloudflare 2025

AuthKEM: Other Approaches for PQ Auth

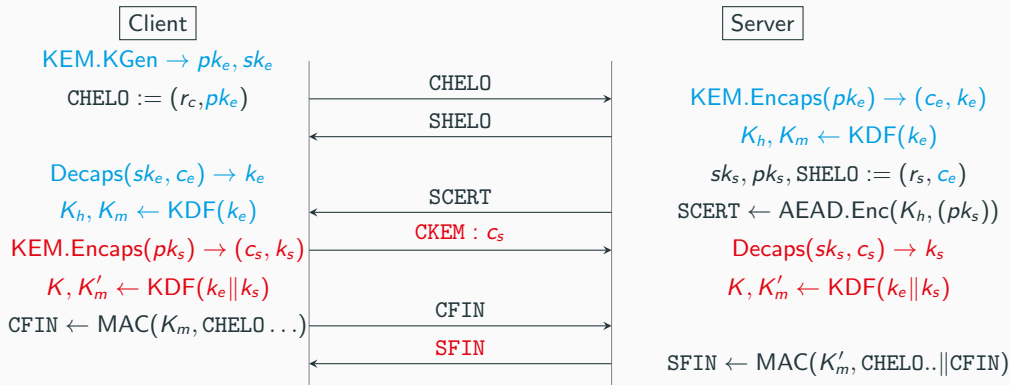
Post-quantum digital signatures are **bandwidth-expensive** compared to KEM ciphertexts. Could we **repurpose KEMs** to provide **authentication guarantees**?

Core idea: Encapsulate a (fresh) secret under long-term KEM key, then use a MAC/key confirmation message to **verify that communicating partner has the associated secret key**.

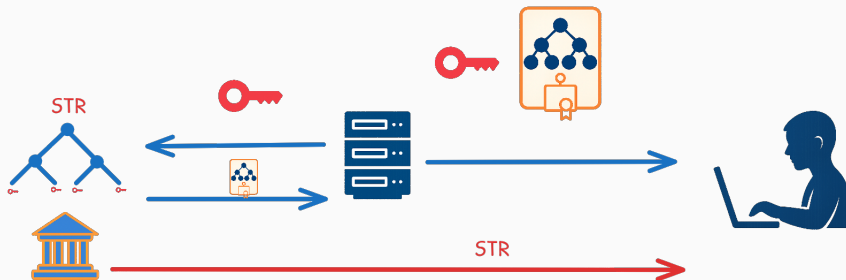


Impact: Imposes extra round trips, since KEMs, unlike signatures, are **interactive**.

Ephemeral KEX KEMs vs static Auth KEMs.

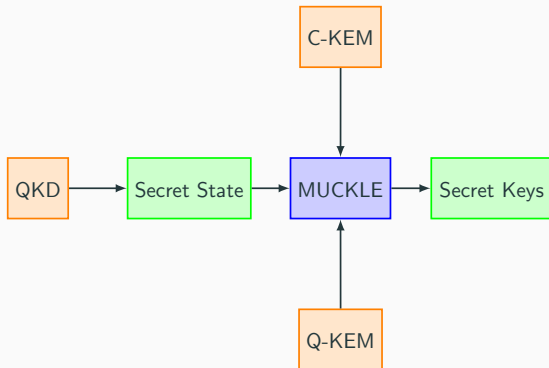


Lessons from TLS's PQ Transition?



- For HNDL hybrid approach favoured: **add, don't replace**
 - How to **combine KEMs and DH**? Many KEM combiners.
- Approaches on **reducing bandwidth** from PQ:
 - **Authenticating via KEMs** reduces “signature size”.
 - Use of **Merkle Tree Certificates** reduces number of signatures.
 - Significant engineering effort to come.

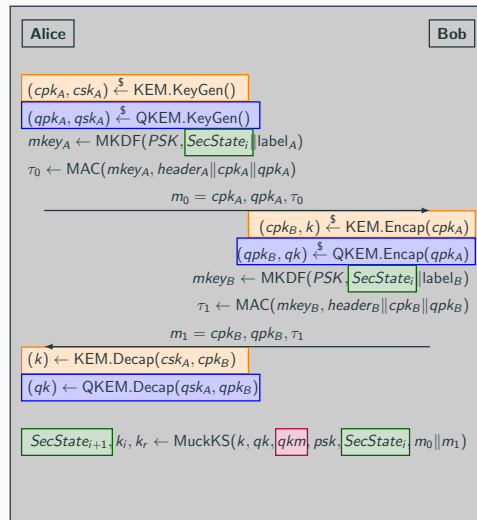
Many a Mickle Makes a Muckle



Combine QKD and PQC to achieve **composition** of security guarantees, even when parts of the system are undermined or **breakdown**

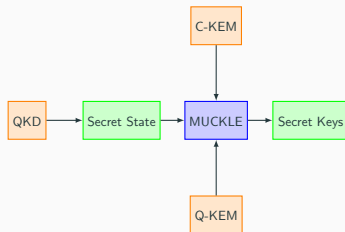
Muckle Overview

- Multi-stage 1-RTT protocol.
- Hybrid KEM achieves breakdown resilience in CKEM, QKEM, QKD components
- Evolving state (*SecState*) achieves PCS
- Use of MACs to achieve PQ auth.
- Modular design allows easy replacement of KEM components.
- Key confirmation and full message transcript agreement of previous stages are provided in successive stages via the computation of authentication keys.



High-level Properties

- **Quantum-Safe KEX:** PQ-KEM, QKD
 - Use ITS KDF for **everlasting security**
- **Quantum-Safe Auth:** PSK and MAC
 - Use ITS MAC for **IT security**
- **Breakdown resilient:** If PQ-KEM, QKD are broken, C-KEM ensures classic security
- **Forward Secrecy:** Use PFS KEMs
- **Post-Compromise:** Evolving *SecState*



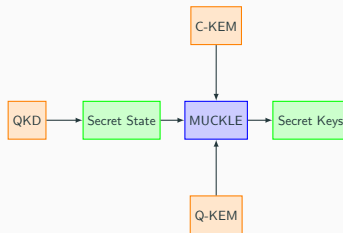
QKD ensures security in face of **PQC vulnerabilities**, PQC in face of **QKD hardware failures**.

Everlasting Security ensures that ITS holds if QC don't exist at time of QKD exchange.

Challenges

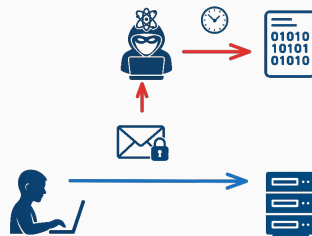
Many variants of Muckle Protocol now exist, addressing the following challenges:

- **Syncing:** PQ-KEM, QKD
 - keys output at different rates, how to combine?
- **Auth:** PSK and MAC
 - PSKs not scalable, how to bootstrap signatures?
- **Loss of ITS:** Need to replace MAC, KDF with ITS variants to achieve
- **Composition:** What are we using this for? What composition guarantees hold?



Status and Lessons Learnt?

- Broad movement towards addressing HNDL attacks with inclusion of **hybrid key-exchange**.
 - Diffie-Hellman's NIKÉ **difficult to replace**, but easy for most cases (ephemeral-ephemeral Diffie-Hellman)
- Post-Quantum **authentication lagging**:
 - If interactive, can use **KEMs** for authentication
- PQ transition for most widespread protocols ongoing
 - Lots of different approaches, **no one-size-fits-all solution**
 - Would be good to develop unified approach to PQ transition.
- Lots of opportunities to change the infrastructure for post-quantum transition!



Thanks, and Questions?

