2021/03/29

CLOUDFLARE

KEMTLS: securing TLS connections from quantum adversaries

Sofía Celi, Thom Wiggers

The People

KEMTLS?

- A protocol that we have been developing in the last months
- We will talk around what it is, why it is needed and how we are experimenting with it.

What is a quantum adversary?

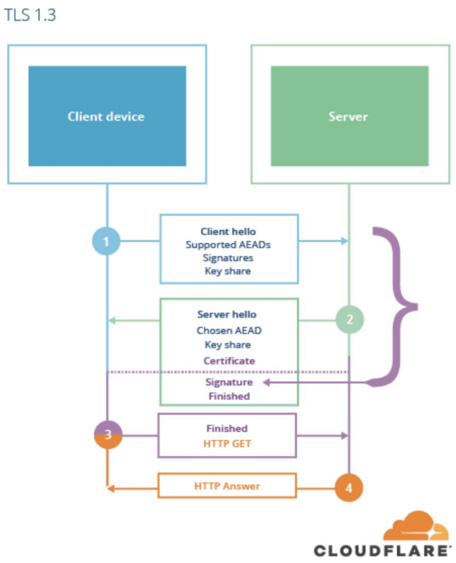
How all started?

- Richard Feynman: a computational model that obeys quantum mechanics.
- They are more efficient but they will break most cryptographic algorithms.
- Shor's and Grover's algorithms solve factorization.
- What can we do? Post-quantum cryptography for signatures and encryption.
- Google/IBM building these machines.

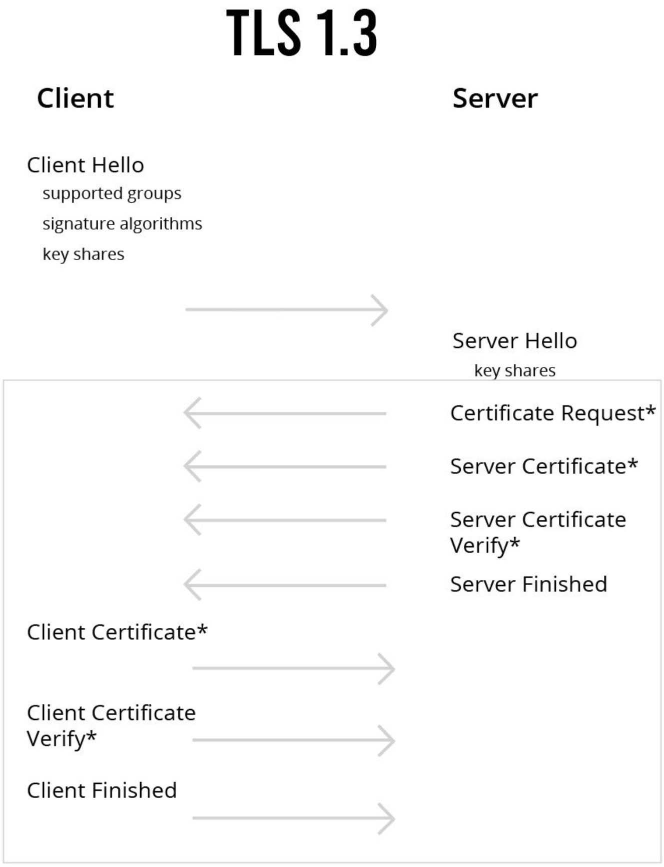
How does TLS work?

Why TLS 1.3?

- TLS?
- It took 5 years for the current version to be made
- TLS 1.3 Goals
  - Achieve certain properties
  - Be as efficient as it can be
  - Encrypting parts of the handshake
  - Improving resilience to certain attacks
- OPTLS?
- How it changed the game?



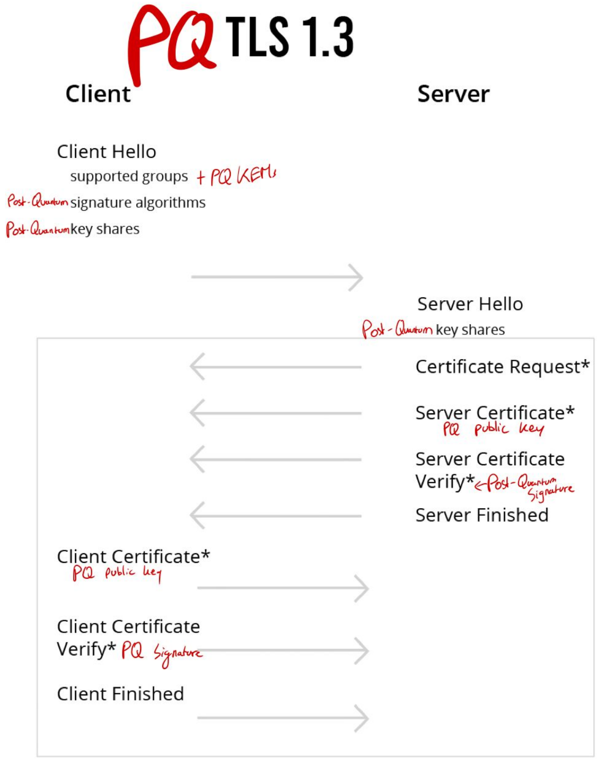
KEMTLS: securing TLS connections from quantum adversaries



- It provides:
  - Authentication
  - Confidentiality
  - Integrity
- When can application data be sent?
- Downgrade resilience

Making TLS Post-Quantum

PQ TLS



PQ TLS 1.3

Client

Server

Client Hello
supported groups + PQ KEMs
Post-Quantum signature algorithms
Post-Quantum key shares

Server Hello
Post-Quantum key shares

Certificate Request*

Server Certificate*
PQ public key

Server Certificate Verify* ←Post-Quantum Signature

Server Finished

Client Certificate*
PQ public key

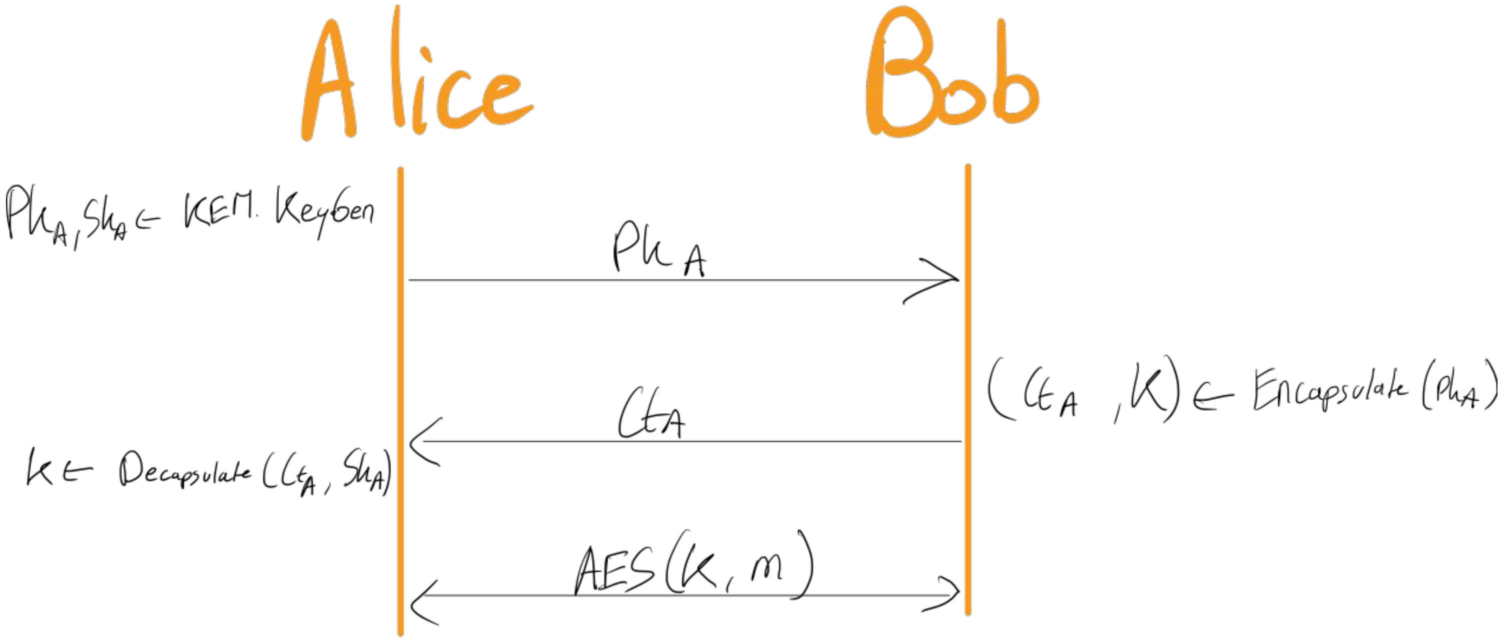
Client Certificate Verify* PQ Signature

Client Finished

A more efficient handshake: KEMTLS

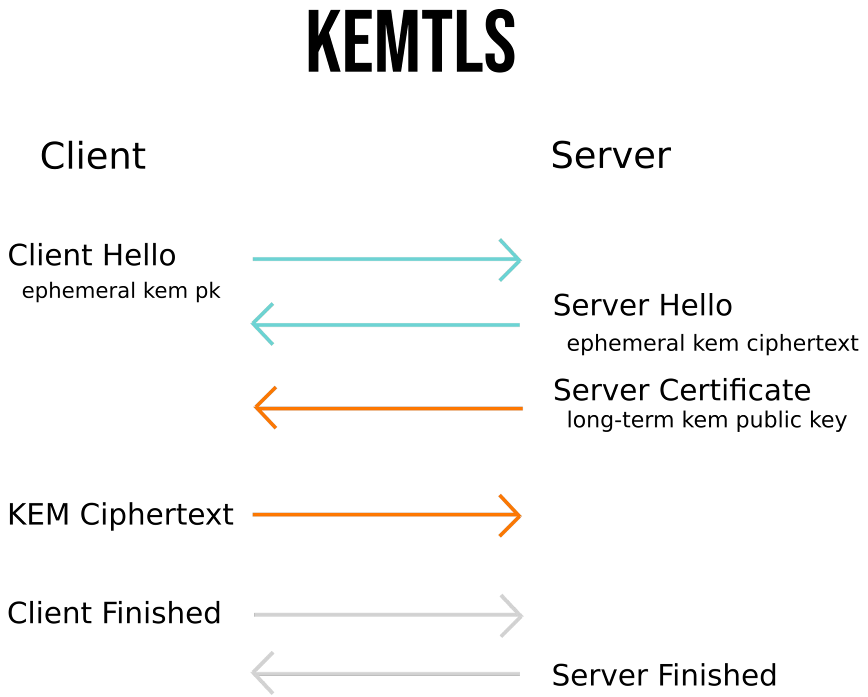
What is a Key Encapsulation Mechanism (KEM)?

- Key exchange


Alice
Bob
Pk_A, Sk_A ← KEM. KeyGen
Pk A
(Ct_A , K) ← Encapsulate (Pk_A)
Ct_A
K ← Decapsulate (Ct_A , Sk_A)
AES(K, m)

KEMTLS

- Ephemeral KEM Key exchange
- Authentication via KEM
- Client can send encrypted data in its first reply



KEMTLS

Client

Server

Client Hello
ephemeral kem pk

Server Hello
ephemeral kem ciphertext

Server Certificate
long-term kem public key

KEM Ciphertext

Client Finished

Server Finished

KEMTLS performance gains

- Kyber512 handshake data size: public key + ciphertext
  - 1568 bytes
- Dilithium2 handshake data size: public key + signature
  - 3732 bytes

- PQTLS: ephemeral key exchange + handshake signature
  - 1568 + 3732 = 5300 bytes
- KEMTLS: ephemeral key exchange + authentication key exchange
  - 1568 + 1568 = 3136 bytes
  - Only 59% as much data!
  - KEM operations typically computationally cheaper than signing

Note: you still need to send and verify a signature chain (CA certificates and signatures)

The Experiments

Experiments

- Past experiments: Cloudflare and Google
- Run KEMTLS over *drand* (distributed randomness beacon) connections with Delegated Credentials
- What we want to compare:
  - TLS 1.3
  - PQTLS
  - Hybrid TLS 1.3
  - KEMTLS
- What we still need: eliminate the "extra" trip
- Add all of the other TLS 1.3 extensions

Thank you!