Quantum-safe hybrid handshake for TLS 1.3

Recent updates

Sep 2015

Zhenfei Zhang
Security Innovation

QSH_TLS

- Run a classical key exchange as usual and obtain a classical pre-master secret *c*;
- In parallel, transport another pre-master secret *q* using a key encapsulation mechanism (KEM), instantiated with a quantum-safe (a.k.a post-quantum) encryption algorithm;
- The final master secret will be derived from KDF($c|q$)

Features

- Defeat the harvest-then-decrypt attack with low cost;
- Modular design allows for trial use of quantum-safe cryptography;
- Requires one additional cipher suite identifier;
  - It would be nicer if no identifier is added.

QSH_TLS

- Run a classical key exchange as usual and obtain a classical pre-master secret *c*;
- In parallel, transport another pre-master secret *q* using a key encapsulation mechanism (KEM), instantiated with a quantum-safe (a.k.a post-quantum) encryption algorithm;
- The final master secret will be derived from KDF(*c*|*q*)

The story so far

- At IETF 93, William Whyte presented the Quantum-Safe Hybrid (QSH) hand shake for TLS 1.3
- No objections to continuing to investigate this approach within TLS but defer to CFRG on algorithm selection
- CFRG hummed unanimously to pursue further investigations of quantum-safe crypto

Updates #0 (Global): QS crypto

- There is a growing concern on quantum safety in the past few months:
  - NSA has advised people to move away from ECC and announced their plan to migrate to quantum-safe cryptography;
    - https://www.nsa.gov/ia/programs/suiteb_cryptography/
  - The EU has expressed in their Horizon 2020 project a desire for systems to be "quantum-ready" by 2020;
    - http://pqcrypto.eu.org/slides/20150403.pdf
  - Google have optimistically predicted practical and powerful quantum computer could become available by the 2020 to 2025.
    - http://www.theplatform.net/2015/07/22/google-sees-long-expensive-road-ahead-for-quantum-computing/
  - More is coming…

Updates #1 (CFRG): algorithm selection

- We have an internet-draft describing the selection criteria of quantum-safe encryption algorithm to be adopted in the QSH_TLS
  - The idea is to
    - setup a base line for QS encryption schemes;
    - provide a list of existing QS encryption schemes meeting those criteria;
    - allows a clear pathway to adoption for future QS schemes.
- CFRG is currently reviewing this document
- (Most of) our initial recommendations align with PQCRYPTO's initial recommendations and ETSI ISG-QSC's recommendations
  - http://pqcrypto.eu.org/docs/initial-recommendations.pdf

Updates #2 (TLS): Cipher Suite -> Extension

- We move QSH_TLS data from KeyShare message to HelloExtension and HelloRetryRequestExtension
  - Following on comments from DKG and others at Prague meeting
- We require an extra ExtensionType, rather than a cipher suite identifier
- The latest version:
  - https://www.ietf.org/internet-drafts/draft-whyte-qsh-tls13-01.txt
- Change made only in TLS 1.3 version of spec, can be propagated into TLS 1.2 version if useful
  - TLS 1.2 version still uses Cipher Suite approach

Updates #3: Performance analysis

- Feedback from ETSI ISG-QSC group
- An additional KEM is likely to increase the cost
  - Latency, not significantly.
    - See table on the right
  - Handshake packet size, may be affected: need a much larger extension field
    - See next slide
- The KDF is not going to add extra cost
  - Previous we do KDF(c)
  - Now we do KDF(c|q)

	Classical strength	Time
NTRU449 encryption	128 bits	2
NTRU743 encryption	256 bits	4.4
RSA2048 decryption	112 bits	100
curve25519 DH	128 bits	3.4

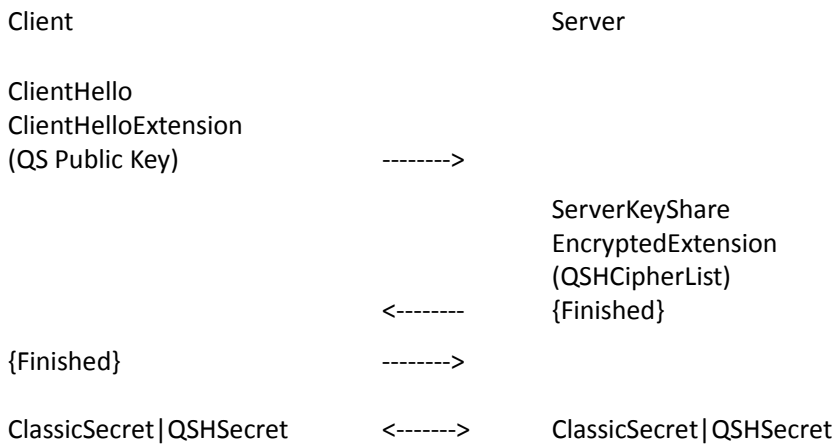
Relative cost on server side
Benchmark from SUPERCOP
http://bench.cr.yp.to/supercop.html

Obstacles #1

- Extension field is limited to 2^16-1 bytes
- This would forbid the use of QS encryption schemes with key/cipher size > 65KB
  - lattice-based crypto are okay, including NTRUEncrypt, R-LWE, etc;
  - code-based crypto are not, including McEliece, McBits, etc;
    - Those keys/ciphertexts are on the order of MB
- We had similar issue with Tor cell size – get away with a “multi cell” solution;
  - This does not work for TLS 1.3
  - There's also an explicit MUST NOT clause for passing multiple extension fields of the same type.
- Proposal: Consider increasing the size limitation on extension fields to, say 2^24-1 byte?

Obstacles #2

- Extension field of KeyShare message is encrypted
  - We could in principle encrypt QSH_TLS message, but that would be redundant
    - The actual data in the QSH_TLS message is a ciphertext of the QS scheme
  - We would request QSH_TLS message to be on the non-encrypt whitelist
    - If TLS WG choose to go along with the whitelist method



Actions

- Extension size limitation?
- Whitelist?
- To get the individual draft adopted as a WG draft
  - The approach is so modular that it doesn’t rely on any QS scheme
  - So we should start working on this draft while CFRG is still considering QS candidates