Planning for Post-Quantum Cryptography

CA/Browser Forum 64, Tokyo, Japan

Russ Housley
Vigil Security, LLC

Motivation

- If Cryptographically Relevant Quantum Computers (CRQCs) are ever built, these computers will be able to break the public key cryptosystems that we currently use.
- A post-quantum cryptography (PQC) is secure against CRQCs.
- It is open to conjecture when it will be feasible to build such quantum computers; however, RSA, DSA, ECDSA, DH, ECDH, and EdDSA are all vulnerable if a CRQC is developed.
- We need to plan for a transition to PQC algorithms.

NIST Hash-based Signature Algorithms

- The U.S. National Institute of Standards and Technology (NIST) has already approved two PQC hash-based signature algorithms and published their specifications: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-208.pdf
- Digital Signatures:
  HSS/LMS (RFC 8554) and XMSS (RFC 8391)

Note: NIST adopted these two algorithms that were already documented in RFCs.

NIST PQC Competition Winners

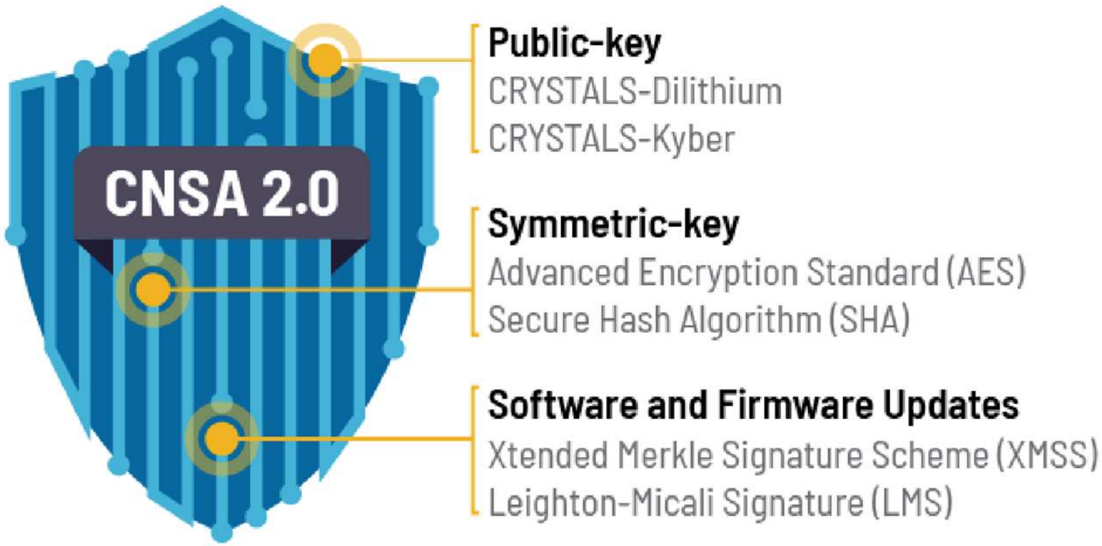
- Key Encapsulation Mechanism (KEM):
  CRYSTALS-KYBER → ML-KEM (FIPS 203)
  HQC → No NIST name yet (not published yet)

- Digital signatures:
  CRYSTALS-DILITHIUM → ML-DSA (FIPS 204)
  SPHINCS+ → SLH-DSA (FIPS 205)
  FALCON → FN-DSA (not published yet)

NSA Announced Direction

About a month after NIST announced the winning algorithms, NSA announced that National Security Systems should begin planning to implement:

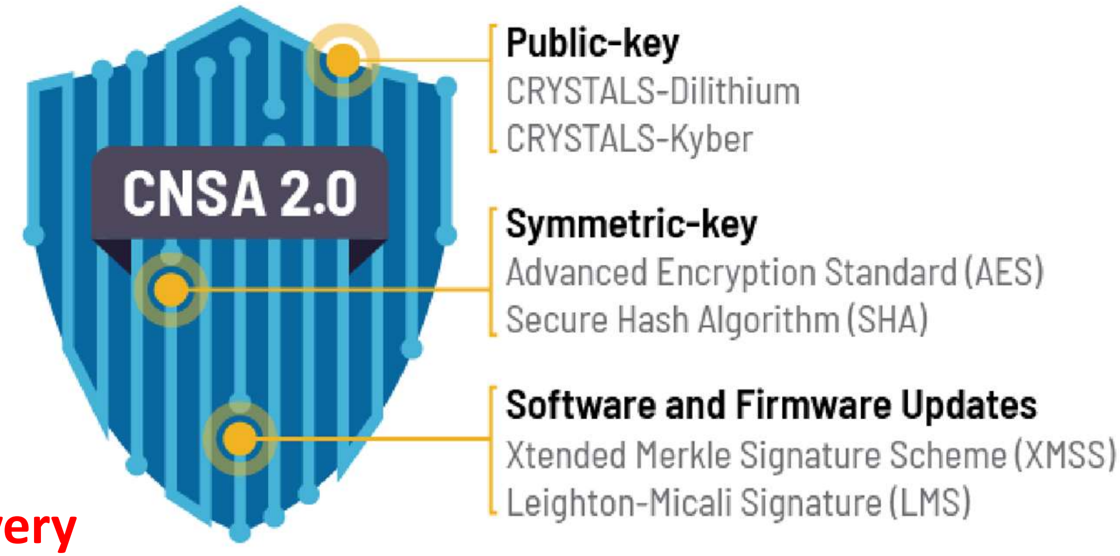
- Prefer HSS/LMS for software signing
- Prefer ML-DSA for other signing
- Prefer ML-KEM for key management



NSA Announced Direction

About a month after NIST announced the winning algorithms, NSA announced that National Security Systems should begin planning to implement:

- Prefer HSS/LMS for software signing
- Prefer ML-DSA for other signing
- Prefer ML-KEM for key management



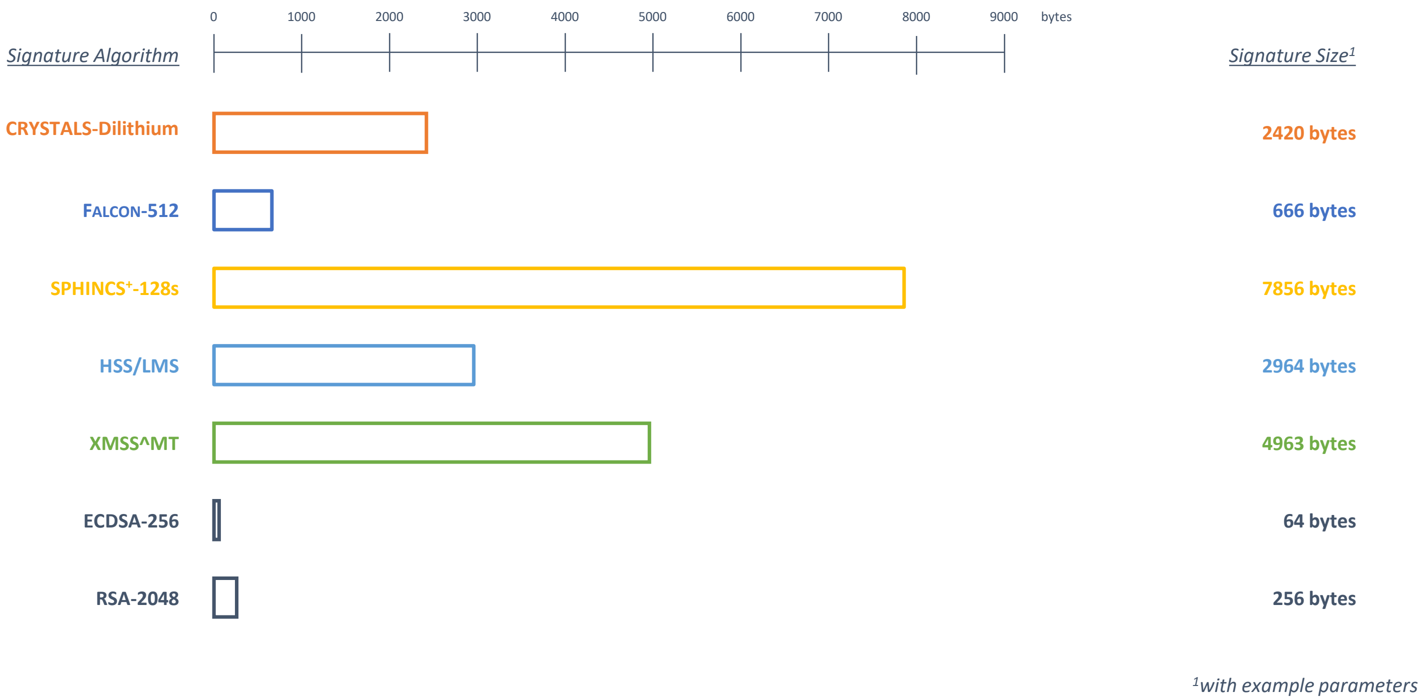
Transition is going to take a very long time. Let's get started!

IETF Security Protocols

Many security protocols are used in the Internet;
all need to support PQC:

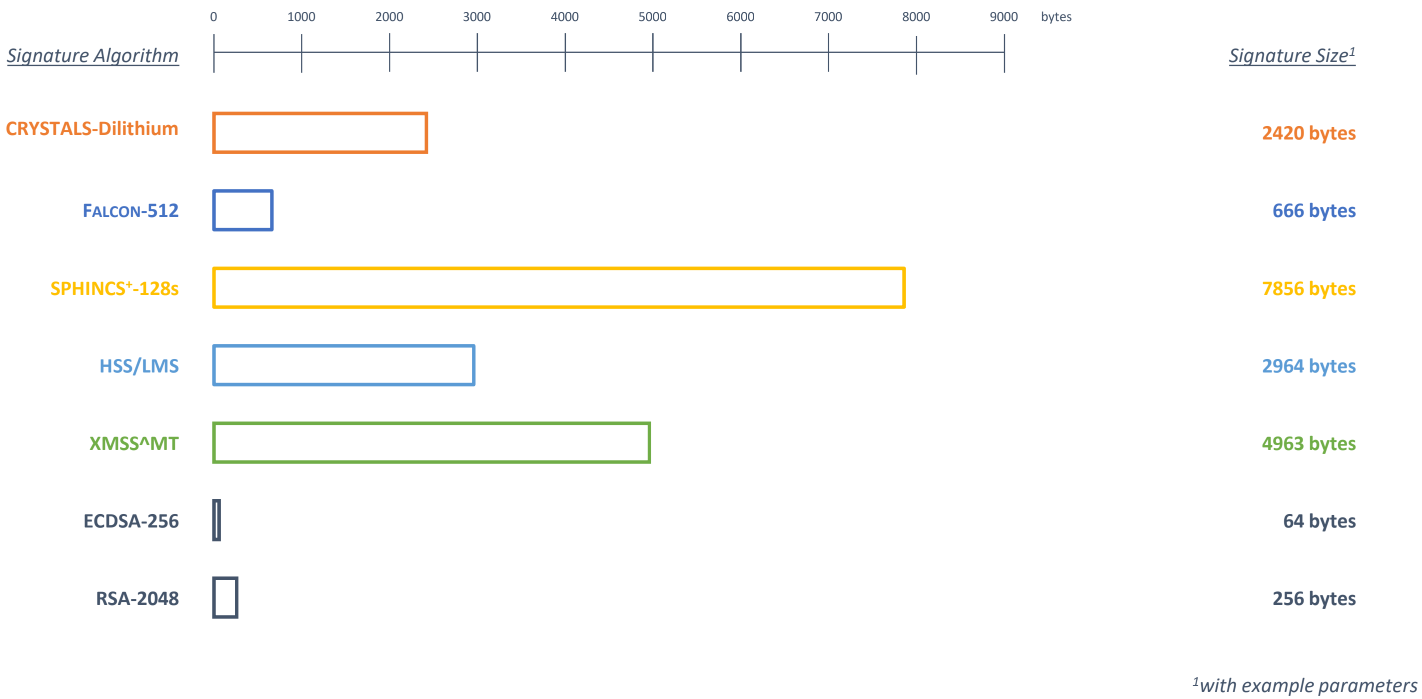
- IPsec
- TLS
- SSH
- S/MIME
- OpenPGP
- …
- Internet profile for X.509 certificates

Large Public Key and Signature Size



Many thanks to VeriSign for this graph

Large Public Key and Signature Size



Plan for an increase of 10X in protocols …

Transition Priorities

Confidentiality – The attacker can record today's traffic, and then break it when a CRQC is eventually developed. (Harvest Now, Decrypt Later.)

Authentication – Tends to be real-time interaction, so not a concern until a CRQC is imminent.

Signature – Tends to be archival, so a notary or archivist can resign with a PQC signature at some point before a CRQC is available. (See RFC 4998: Evidence Record Syntax.)

PQC Algorithms and Certificates

Goal – Deploy PQC algorithms before a CRQC that is available to break the public key algorithms in widespread use today.

Assumption – While people gain confidence in the PQC algorithms and their implementations, security protocols will mix traditional algorithms and PQC algorithms.

Recognize – Such transitions take a long time—at least a decade.

Timeframe – NIST recommends PQC infrastructure in place by 2030, and stop using traditional public key algorithms by 2035.

Note: People that trust the PQC algorithms can transition without mixing with a traditional public key algorithm. However, a lengthy transition is still needed.

Transition History: SHA-1 to SHA-256

- NIST recommended end of life for SHA-1 by end of 2010
- IETF started work in 2005 – 5 years should be enough!
- In the same year, Wang showed that SHA-1 was not as strong as expected
  - 2^{69} instead of 2^{80} design goal
  - Subsequent research improved the attack, reduced strength to 2^{63}
  - Additional incentive!
- **The transition still took more than 10 years**
- NIST goal is less than 10 years to stop using traditional public key

Two Possible Certificate Approaches

Hybrid: Two certificates, each certificate has one public key and one signature:

- one certificate traditional algorithm, signed with traditional algorithm
- one certificate PQC algorithm, signed with PQC algorithm

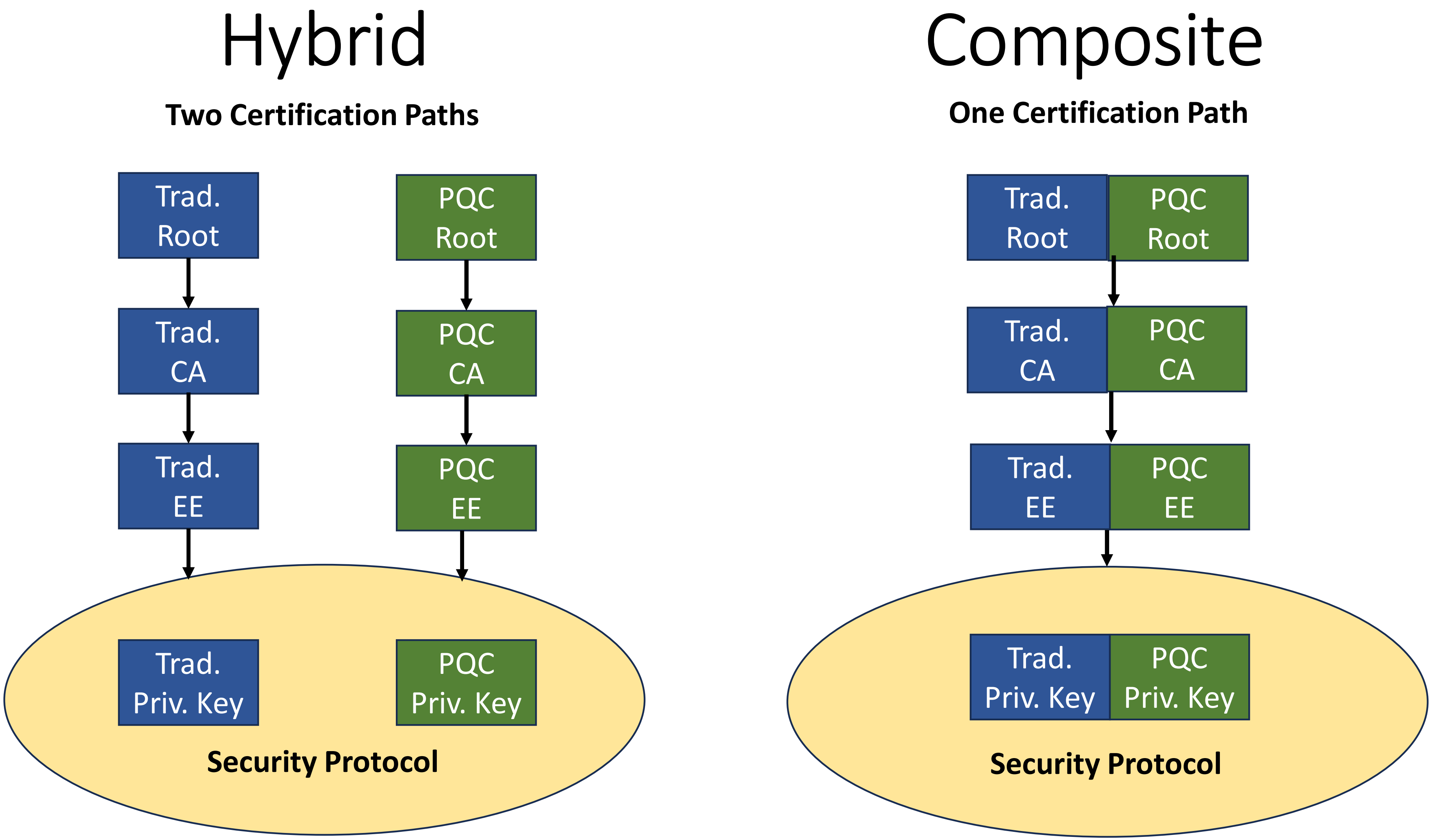
Composite: One certificate, containing two public keys and two signatures:

Public Key

SEQUENCE OF	Traditional public key
	PQC public key

Signature

SEQUENCE OF	Traditional signature
	PQC signature


Hybrid
Two Certification Paths
Trad. Root
PQC Root
Trad. CA
PQC CA
Trad. EE
PQC EE
Trad. Priv. Key
PQC Priv. Key
Security Protocol
Composite
One Certification Path
Trad. Root
PQC Root
Trad. CA
PQC CA
Trad. EE
PQC EE
Trad. Priv. Key
PQC Priv. Key
Security Protocol

Gaining Confidence (session-oriented)

- While people gain confidence in the PQC algorithms and their implementations, security protocols are expected to mix traditional and PQC algorithms

- IPsec and TLS, use a KDF to compute shared secret from two inputs:

$$\mathbf{SS = KDF(\ SS_{T},\ SS_{PQC}\ )}$$

Gaining Confidence (session-oriented)

- While people gain confidence in the PQC algorithms and their implementations, security protocols are expected to mix traditional and PQC algorithms

- IPsec and TLS, use a KDF to compute shared secret from two inputs:

$$\mathbf{SS = KDF(\ SS_T,\ SS_{PQC}\ )}$$

For example:

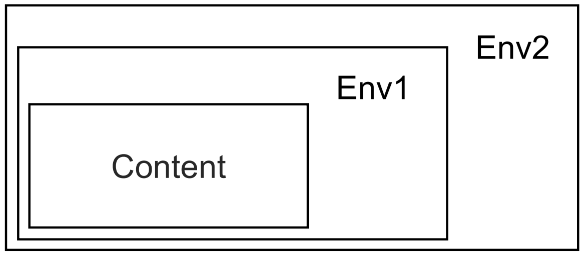
ML-KEM

Diffie-Hellman

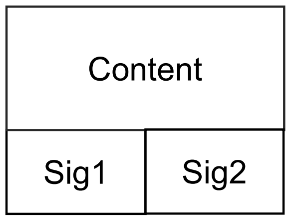
Gaining Confidence (store and forward)

S/MIME could do the same as IPsec and TLS, *or* more likely, S/MIME use double encapsulation:

Encryption



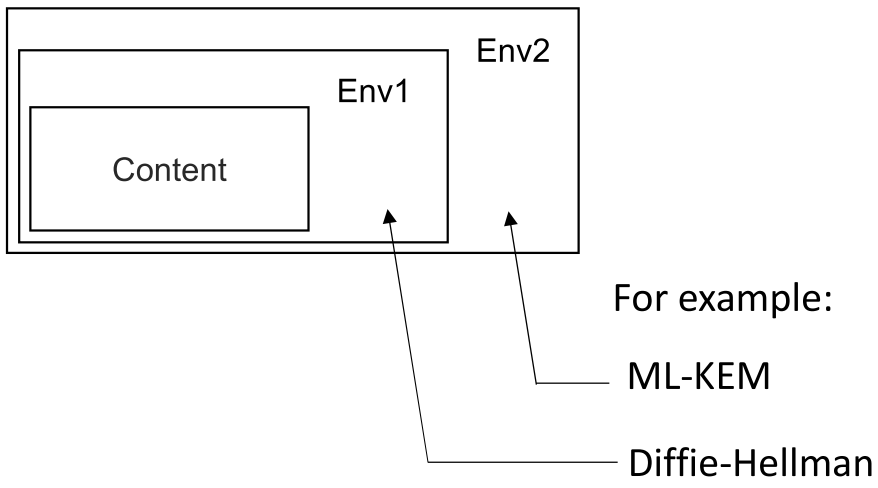
Signature



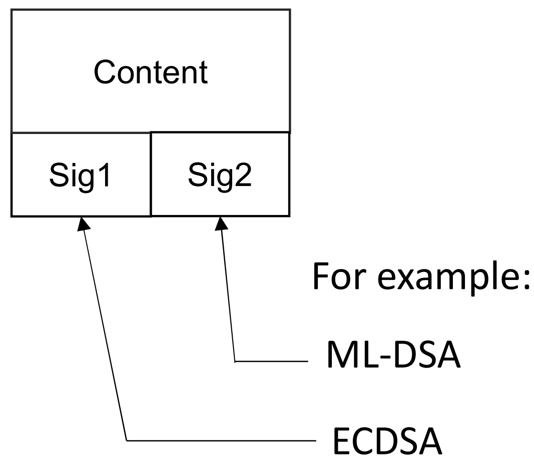
Gaining Confidence (store and forward)

S/MIME could do the same as IPsec and TLS, *or* more likely,
S/MIME use double encapsulation:

Encryption



Signature



IETF SUIT Working Group

The IETF SUIT WG has specified a signed manifest for software updates. A PQC signature will be one of the mandatory to implement algorithms:

- Signing the software with a PQC algorithm offers a way to deploy other PQC algorithms, even if a CRQC is invented soon
- Current draft specification requires implementation of HSS/LMS

IETF IPsecME Working Group

The IETF IPsecME WG has already specified a way for IKEv2 peers to perform multiple successive key exchanges:

- **IKE_SA_INIT**: Always a traditional algorithm
- **IKE_INTERMEDIATE**: Allows PQC algorithms, and supports message fragmentation to handle the large public key sizes
- If any of the key exchange methods is a PQC algorithm, then the final keying material is post-quantum secure
- IPsecME WG is specifying the NIST PQC algorithms for IKEv2

IETF TLS Working Group

The IETF TLS WG is defining the *hybrid* key exchange, which uses two or more algorithms to produce a final session key that is secure as long as at least one of the component key exchange algorithms remains unbroken.

- Client and server send the key shares, then they construct the **concatenated_shared_secret** by:
  shared_secret_1 || shared_secret_2 || … || shared_secret_n
- Compute the Handshake Secret in the TL 1.3 key schedule:
  concatenated_shared_secret -> HKDF-Extract = Handshake Secret
- TLS WG is specifying the hybrid and pure PQC algorithms for TLS 1.3

IETF LAMPS Working Group

The IETF LAMPS WG is specifying pure NIST PQC algorithms and composite algorithms for both certificates and S/MIME:

- specify the use of the NIST PQC public key algorithms using the object identifiers that are assigned by NIST
- specify formats, identifiers, enrollment, and operational practices for hybrid key establishment algorithms
- specify formats, identifiers, enrollment, and operational practices for dual signature algorithms

IETF OpenPGP Working Group

The IETF OpenPGP WG is specifying:

- the use of the pure SLH-DSA for digital signature
- composite public-key encryption based on ML-KEM and two elliptic curve algorithms (X25519, X448)
- composite public-key signatures based on ML-DSA and EdDSA

IETF SSHM Working Group

The IETF SSHM (Secure Shell Maintenance) WG is specifying:

- composite public-key encryption based on ML-KEM and ECDH
- composite public-key encryption based on Streamlined NTRU Prime[1] (not a standard) and X25519

[1] https://ntruprime.cr.yp.to/nist/ntruprime- 20201007.pdf

Any Questions