# Levels of standardization

<table>
<thead>
<tr>
<th>Use in protocols</th>
<th>• IETF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formats and identifiers</td>
<td>• IETF</td>
</tr>
</tbody>
</table>
| Algorithms        | • IRTF CFRG  
|                   | • NIST |

Standardizing PQ algorithms
PQ algorithms being standardized

Lattice-based KEMs
- Kyber
- NTRU

Lattice-based signatures
- Dilithium
- Falcon

Stateful hash-based signatures
- XMSS
- HSS/LMS

Stateless hash-based signatures
- SPHINCS+
Trade-offs with post-quantum crypto

Confidence in quantum-resistance

Fast computation

Small communication

Pick ~2
Trade-offs with post-quantum crypto

**rsa and elliptic curves**
- Fast computation
- Small communication
- Confidence in quantum-resistance

**lattice-based cryptography**
- Fast computation
- Small communication
- Confidence in quantum-resistance

**hash-based signatures**
- Fast computation
- Small communication
- Confidence in quantum-resistance

TLS handshake:
- RSA and elliptic curves: 1.3 KB
- Lattice-based cryptography: 11.2 KB
- Hash-based signatures: 24.6 KB
Challenges
putting PQ cryptography into protocols
Challenge: larger communication sizes

Higher bandwidth usage
- Impact on high-traffic providers
- Higher power usage in battery-operated devices

Higher latency
- Larger data in early flows of TCP leads to more round trips if exceeding the TCP congestion window
- More packets on poor-quality links leads to more retransmission

Impossible to fit in some protocols
- e.g. DNSSEC over UDP has problems with packets larger than 1232 bytes

Challenge: KEMs ≠ Diffie–Hellman

$(pk, sk) \leftarrow \text{KEM.KeyGen}()$

$pk$ \hspace{2cm} \text{Responder's operation depends on sender's pk value}$

$(ct, k) \leftarrow \text{KEM.Encaps}(pk)$

$ct$ \hspace{2cm}$k \leftarrow \text{KEM.Decaps}(sk, ct)$

- Can't achieve non-interactive key exchange
- Can't achieve certain types of authenticated key exchange handshake flows
- Problems instantiating other DH-based primitives
  - ZK, OPRFs, …
Challenge: state in stateful hash-based sigs.

Unsafe to reuse portions of secret key in stateful hash-based signatures

- Need to manage state across all devices signing using the same key
  - "The cryptographic module shall not allow for the export of private keying material." [1]

- Need to worry about key exhaustion
  - Establish a limit on number of signatures at key generation time
  - We have parameter sets that can sign 1 trillion times, but with 1.5 hour keygen time

[1] NIST SP 800-208 Recommendation for Stateful Hash-Based Signature Schemes, Section 8.1
Challenge: many options to choose from

**KEMs**

- **Kyber**: 1 option for each of 3 security levels

- Pure PQ?
- Hybrid?
  - With what elliptic curves?
  - With what combiner?

**Signatures**

- **Dilithium**: 1 option for each of 3 security levels
- **Falcon**: 1 option for each of 2 security levels
- **SPHINCS+**: 4 options for each of 3 security levels
- **XMSS**: 44 options across 3 security levels
- **LMS**: 9+15 options

- Pure PQ?
- Hybrid?
  - With RSA at what level?
  - With what elliptic curve?
- Certificate chain: different algorithms for root/intermediate?
Addressing the challenges of using PQ crypto

- Lack of confidence in security
- Slow computation
- Large communication
- KEMs ≠ DH
- State in hash-based signatures
- Many options

"Just" make better PQ crypto!
Addressing the challenges of using PQ crypto

- Lack of confidence in security
  "Hybrid": Use multiple algorithms

- Slow computation
  Actually not too bad?

- Large communication
  Accept it; or change how security and network protocols use PQ

- KEMs ≠ DH
  Do what we can now; open research questions

- State in hash-based signatures
  Be careful

- Many options
  Standards bodies make the tough choices

- Standards bodies make the tough choices
Hybrid / composite
Hybrid approach: use traditional and post-quantum simultaneously such that successful attack needs to break both
## Hybrid and composite terminology

<table>
<thead>
<tr>
<th></th>
<th>Composite hybrid</th>
<th>Non-composite hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol fields and messages</td>
<td>Unchanged</td>
<td>Changed to accommodate multiple elements</td>
</tr>
<tr>
<td>Cryptographic elements</td>
<td>Combined using a newly defined format</td>
<td>Unchanged</td>
</tr>
</tbody>
</table>

Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm

2. Ease transition with improved backwards compatibility and agility

3. Standards compliance during transition
Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm
2. Ease transition with improved backwards compatibility and agility
3. Standards compliance during transition
   - Early adopters may want to use post-quantum before standards-compliant (FIPS-)certified implementations are available
   - Possible to combine (in a certified way) keying material from certified (non-PQ) implementation with non-certified keying material
Why to **not** use hybrid

- Increases number of design choices
- Increases implementation complexity
- Increases code size

Regulatory fracturing:
- Hybrids required: BSI (Germany), ANSSI (France)
- Hybrids allowed: ENISA (EU), ETSI
- Hybrids discouraged: NSA (US)
  - No decision on hybrids: NCSC (UK), CSE (Canada)

Hybrid key exchange

- Use two (or more) key exchange methods
- Transmit both public keys
- Combine shared secrets using an appropriate mechanism

- Fairly well understood
- Lots of progress in Internet-Drafts and prototypes
- Seems likely to be broadly adopted in first phase of PQ transition
How to combine shared secrets

1. Concatenate & hash [1]
2. NIST-approved methods (SP 800-56C)
   - Concatenate traditional & PQ shared secret
   - Use as input to one-step or two-step KDFs
   - Including HKDF

Is concatenation safe?
- Yes, if H is a random oracle
- Yes, if H is a dual-PRF
- Not necessarily, if H is not collision resistant and secrets are variable length and reused [2]
- More research to be done here

Hybrid authentication

• Use two (or more) authentication methods
• Transmit both public keys and signatures
• Validate both signatures

Discussion to be had on when hybrid authentication is desired

• May be unnecessary in the context of interactive / negotiated protocols and short-term authentication

• May be relevant for long-term scenarios like firmware updates and document signing
  • Counterargument: just use hash-based signatures
Standardizing PQ formats and identifiers
Standardizing PQ formats and identifiers

Single protocol-specific

- Examples:
  - TLS signature algorithm or key exchange "group"
  - SSH method name
- To be done in corresponding working group document
- To be added to IANA registry

Used in multiple protocols

- Examples:
  - Private key export format
  - Algorithm naming
  - ASN.1 Object identifiers
- Multiple drafts for Kyber, Falcon, Dilithium, SPHINCS+ key formats
  - Individual drafts
  - LAMPS/COSE working groups

Caution: early prototypes have ad hoc formats and identifiers

Caution: early prototypes may use incompatible algorithm versions (Round 1, 2, …)
Standardizing PQ use in protocols
Post-quantum TLS
What is “post-quantum TLS”?

**Pre-shared key (PSK) mode**

- Already implemented
- Still has key distribution problem
- No forward secrecy

**Alternative protocol designs**

- PQ-only
- Hybrid
- Composite

- Already implemented
- Fairly easy to implement
- Needed soonest: harvest now, decrypt later
- Requires coordination with certificate authorities
- Less urgently needed: can’t retroactively break authentication

**Examples**

- AuthKEM/KEMTLS: Harder to implement; may require state machine changes
- Lots of interesting research!

- Robust to 1 algorithm break
- Possibly in demand during pre-certification
- May not make sense in the context of a negotiated protocol like TLS
## What is “post-quantum TLS”? 

<table>
<thead>
<tr>
<th>Pre-shared key (PSK) mode</th>
<th><strong>Key exchange</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PQ-only</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Hybrid</strong></td>
</tr>
<tr>
<td>• Already implemented</td>
<td>• Fairly easy to implement</td>
</tr>
<tr>
<td>• Still has key distribution problem</td>
<td>• Needed soonest: harvest now, decrypt later</td>
</tr>
<tr>
<td>• No forward secrecy</td>
<td>• Robust to 1 algorithm break</td>
</tr>
<tr>
<td></td>
<td>• Possibly in demand during pre-certification</td>
</tr>
</tbody>
</table>

- **AuthKEM/KEMTLS**
  - Harder to implement; may require state machine changes
  - Lots of interesting research!
  - Robust to 1 algorithm break
  - Possibly in demand during pre-certification

- **PQ-only**
  - Fairly easy to implement
  - Needed soonest: harvest now, decrypt later

- **Hybrid**
  - Requires coordination with certificate authorities
  - Less urgently needed: can’t retroactively break authentication

- **Hybrid / Composite**
  - Already implemented
  - Still has key distribution problem
  - No forward secrecy
  - Needed soonest: harvest now, decrypt later
  - Fairly easy to implement
What is “post-quantum TLS”?

<table>
<thead>
<tr>
<th>Pre-shared key (PSK) mode</th>
<th>Key exchange</th>
<th>Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PQ-only</td>
<td>Hybrid</td>
</tr>
<tr>
<td></td>
<td>• Fairly easy to implement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Needed soonest: harvest now, decrypt later</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Robust to 1 algorithm break</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Possibly in demand during pre-certification</td>
<td></td>
</tr>
</tbody>
</table>

- Already implemented
- Still has key distribution problem
- No forward secrecy
- May not make sense in the context of a negotiated protocol like TLS
- Fairly easy to implement
- Needed soonest: harvest now, decrypt later
- Requires coordination with certificate authorities
- Less urgently needed: can’t retroactively break authentication
- Size 😊
What is “post-quantum TLS”?

<table>
<thead>
<tr>
<th>Pre-shared key (PSK) mode</th>
<th>Key exchange</th>
<th>Authentication</th>
<th>Alternative protocol designs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PQ-only</td>
<td>Hybrid</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Already implemented</td>
<td>• Fairly easy to implement</td>
<td>• Requires coordination with certificate authorities</td>
<td>• e.g. AuthKEM / KEMTLS</td>
</tr>
<tr>
<td>Still has key distribution problem</td>
<td>• Needed soonest: harvest now, decrypt later</td>
<td>• Less urgently needed: can’t retroactively break authentication</td>
<td>• Harder to implement; may require state machine changes</td>
</tr>
<tr>
<td>No forward secrecy</td>
<td>• Robust to 1 algorithm break</td>
<td>• Size 😍</td>
<td>• Lots of interesting research!</td>
</tr>
<tr>
<td></td>
<td>• Possibly in demand during pre-certification</td>
<td>• May not make sense in the context of a negotiated protocol like TLS</td>
<td></td>
</tr>
</tbody>
</table>

Area of initial focus
Hybrid key exchange in TLS

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 31 August 2023

D. Stebila
University of Waterloo
S. Fluhrer
Cisco Systems
S. Gueron
U. Haifa, Amazon Web Services
27 February 2023

Hybrid key exchange in TLS 1.3
draft-ietf-tls-hybrid-design-06

Abstract

Hybrid key exchange refers to using multiple key exchange algorithms simultaneously and combining the result with the goal of providing security even if all but one of the component algorithms is broken. It is motivated by transition to post-quantum cryptography. This document provides a construction for hybrid key exchange in the Transport Layer Security (TLS) protocol version 1.3.

Discussion of this work is encouraged to happen on the TLS IETF mailing list tls@ietf.org or on the GitHub repository which contains the draft: https://github.com/dstebila/draft-ietf-tls-hybrid-design.

• Fairly mature
• Early deployments showing reasonable performance:
  • Chrome experiments
  • Cloudflare
  • Open Quantum Safe
  • WolfSSL
  • …
• Draft in a holding state pending final version of Kyber by NIST and CFRG
  • Placeholder algorithm identifiers
Post-quantum X.509 certificates

Check out the IETF hackathon on PQC certificates
https://github.com/IETF-Hackathon/pqc-certificates
X.509 certificates – algorithm identifiers

**KEMs**
- Internet-Draft for:
  - Kyber algorithm identifiers (placeholder OID)

**Signatures**
- Internet-Drafts for:
  - Dilithium algorithm identifiers (placeholder OID)
  - Hash-based signature algorithm identifiers and data structures
    - HSS/LMS, XMSS
    - SPHINCS+ (placeholder OID)
Composite design choice: how to identify combinations

**Option #1:**
Generic composite

- Single algorithm id representing “composite”, then an additional field containing list of algorithms
  - Good for prototyping
  - Allows for high degree of agility
  - Allows ≥ 2 algorithms

**Option #2:**
Explicit composite

- New algorithm id for each combination of algorithms
  - Less new processing logic
  - Lower degree of agility
    - Easier to test
  - Combinatorial explosion of identifiers
Composite design choice: are all component algorithms in a hybrid required?

How is a credential with two public keys/signatures meant to be used?

- Must both algorithms be used? (Composite AND)
- Is either algorithm okay? (Composite OR)
  - Must take countermeasures to avoid stripping/separating context
  - Risks of ambiguity
  - Algorithm deprecation tricky to handle
Hybrid in X.509

**Composite keys & signatures**
- Internet-Drafts for:
  - Composite public/private keys
  - Composite signatures
  - Threshold composite signatures
    - K-out-of-N required

**Non-composite signatures**
- Internet-Drafts for:
  - Non-composite authentication (expired)
  - Binding related certificates
PQ in other protocols
Secure Shell (SSH)

Key exchange
- Hybrid KEX Internet-Draft available
  - Multiple implementations (Amazon, OQS, wolfSSH, …)
  - OpenSSH using Streamlined NTRU Prime + x25519 by default since OpenSSH v9 (April 2022)

Authentication
- No Internet-Drafts for authentication
- Experiments:
  - OQS PQ & hybrid auth
  - OpenSSH using XMSS-based authentication since OpenSSH v7.7 (April 2018)
    - (Not compiled in by default)
IPsec / IKEv2

Key exchange
• RFC for pre-shared keys
• Internet-Drafts for
  • Multiple key exchanges
  • Mechanisms for handling large messages

Authentication
• Internet-Drafts for
  • Hybrid non-composite authentication
  • Negotiation of authentication methods
CMS
Cryptographic Message Syntax; used in S/MIME

Key exchange / PKE
• Internet-Draft for:
  • KEMs in CMS and Kyber specifically

• Composite "for free"

Authentication
• RFC for:
  • LMS in CMS

• Internet-Draft for:
  • SPHINCS+ in CMS
DNSSEC

Authentication

• Internet-Drafts for:
  • Stateful hash-based signatures (expired)

Research ideas

• Merkle Tree ladder [1]
• Request-based fragmentation [2]

[1] https://eprint.iacr.org/2022/1730
OpenPGP

Public key encryption
• Internet-Draft for:
  • Composite PQ/T Kyber + elliptic curves

Digital signatures
• Internet-Draft for:
  • Composite PQ/T Dilithium + elliptic curves
  • SPHINCS+ (standalone – non-hybrid)
Alternative protocol designs

**Strategy #1:**
Change cryptographic protocols to use PQ algorithms more cleverly/efficiently

- AuthKEM / KEMTLS [1]
- Merkle Tree certificates [2]

**Strategy #2:**
Change network protocols to be more communication efficient

- Technically about reducing latency due to communication size, not reducing communication size itself
- DNSSEC ARRF [3]
- TurboTLS [4]

Wrapping up
## Algorithm standardization status

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Kyber</th>
<th>Dilithium</th>
<th>Falcon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary standardizer:</td>
<td>NIST</td>
<td>NIST</td>
<td>NIST</td>
</tr>
<tr>
<td>Status at NIST:</td>
<td>Draft standard pending</td>
<td>Draft standard pending</td>
<td>Draft standard pending</td>
</tr>
<tr>
<td>Status at IETF/IRTF:</td>
<td>CFRG draft available</td>
<td>No draft available</td>
<td>No draft available</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SPHINCS+</th>
<th>XMSS</th>
<th>LMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary standardizer:</td>
<td>NIST</td>
<td>IRTF</td>
<td>IRTF</td>
</tr>
<tr>
<td>Protocol</td>
<td>Key exchange / PKE</td>
<td>Authentication</td>
<td>Alternatives</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td><strong>TLS 1.3</strong> (secure channel)</td>
<td>Drafts: Hybrid Kyber</td>
<td>Prototypes</td>
<td>• AuthKEM / KEMTLS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• TurboTLS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Merkle Tree certs.</td>
</tr>
<tr>
<td><strong>X.509</strong> (certificates)</td>
<td>Drafts:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Identifiers for Kyber</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Secure Shell (SSH)</strong> (secure channel)</td>
<td>Drafts: Hybrid Kyber</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenSSH:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Hybrid NTRU Prime</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IPsec</strong> (secure channel)</td>
<td>RFCs: PSK</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Drafts: hybrid, large messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Hybrid non-composite</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Negotiation</td>
</tr>
<tr>
<td><strong>CMS</strong> (secure email, …)</td>
<td>Drafts: KEMs, Kyber</td>
<td>RFCs: LMS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Drafts:</td>
<td>Drafts: SPHINCS+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stateful HBS</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DNSSEC</strong> (Domain Name Security)</td>
<td>Drafts: Stateful HBS</td>
<td></td>
<td>• Merkle Tree ladder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Request-based frag.</td>
</tr>
<tr>
<td><strong>OpenPGP</strong> (secure email)</td>
<td>Drafts: Composite Kyber</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timeline to replace cryptographic algorithms

- 1995: SHA-1 standardized
- 2001: SHA-2 standardized
- 2005: SHA-1 weakened
- 2017 Jan.: First full collision for SHA-1
- 2017 Aug.: Browsers stop accepting SHA-1 certificates
- 2023: PQ Final standard
- 2024?: PQ Final standard
Research questions

Being aware of real-world constraints can spawn interesting research questions

Observation: PQ signatures are bigger than PQ KEMs keys
⇒ Implicitly authenticated key exchange would be smaller
⇒ How to make TLS implicitly authenticated?
⇒ Paper on KEMTLS

Observation: X.509 certificates might contain KEM public keys
⇒ Certificate authorities demand proof-of-possession of public keys
⇒ Users like non-interactive proof of possession (certificate signing requests)
⇒ How to do non-interactive proof of possession of KEM public keys?
⇒ Paper on non-interactive proof of Kyber and FrodoKEM keys
Levels of standardization:
• Algorithms: NIST, IRTF CFRG
• Formats and identifiers: IETF
• Use in protocols: IETF

Trade-offs and challenges:
- Lack of confidence in security
- Large communication
- State in hash-based signatures
- Slow computation
- KEMs ≠ DH
- Many options

IETF/IRTF activities:
• Algorithms: Draft for Kyber, RFCs for XMSS & LMS
• Protocols:
  • TLS: Draft for hybrid KEX; several prototypes; alt. designs
  • SSH: Draft for hybrid KEX; shipping in OpenSSH
  • IPsec, CMS, DNSSEC, OpenPGP
• Working groups:
  • TLS, LAMPS, …
Open Quantum Safe Project

Use in applications
- Apache httpd
- nginx
- curl, links
- OpenVPN
- Chromium

Integration into forks of widely used open-source projects

C language library, common API
- x86/x64 (Linux, Mac, Windows)
- ARM (Android, Linux)

liboqs

key exchange / KEMs
- isogenies
- code-based
- lattice-based

signatures
- multi-variate polynomial
- hash-based / symmetric

OpenSSL
S/MIME, TLS 1.3, X.509
OpenSSL 3 provider

BoringSSL

Open SSH

Language SDKs
C#, C++, Go, Java, Python, Rust

Integration into forks of widely used open-source projects

Use in applications

C language library, common API
- x86/x64 (Linux, Mac, Windows)
- ARM (Android, Linux)

Open Quantum Safe Project


Led by University of Waterloo

Industry partners:
- Amazon Web Services
- Cisco
- evolucionQ
- IBM Research
- Microsoft Research

Additional contributors:
- Senetas
- PQClean project
- Individuals

Financial support:
- AWS
- Canadian Centre for Cyber Security
- Cisco
- NLNet
- NSERC
- Unitary Fund
- Verisign
Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm
   - Enable early adopters to get post-quantum security without abandoning security of existing algorithms
   - Retain security as long as at least one algorithm is not broken
   - Uncertainty re: long-term security of existing cryptographic assumptions
   - Uncertainty re: newer cryptographic assumptions

2. Ease transition with improved backwards compatibility

3. Standards compliance during transition
Why use two (or more) algorithms?

1. Reduce risk from break of one algorithm

2. Ease transition with improved backwards compatibility
   - Design backwards-compatible data structures with old algorithms that can be recognized by systems that haven't been upgraded, but new implementations will use new algorithms
   - May not be necessary for negotiated protocols like TLS

3. Standards compliance during transition
On **fast, reliable network links**, the cost of public key cryptography dominates the median TLS establishment time, but does not substantially affect the 95th percentile establishment time.

On **unreliable network links** (packet loss rates ≥ 3%), communication sizes come to govern handshake completion time.

As application data sizes grow, the relative cost of TLS handshake establishment diminishes compared to application data transmission.