Standardizing Post-Quantum Cryptography at the IETF

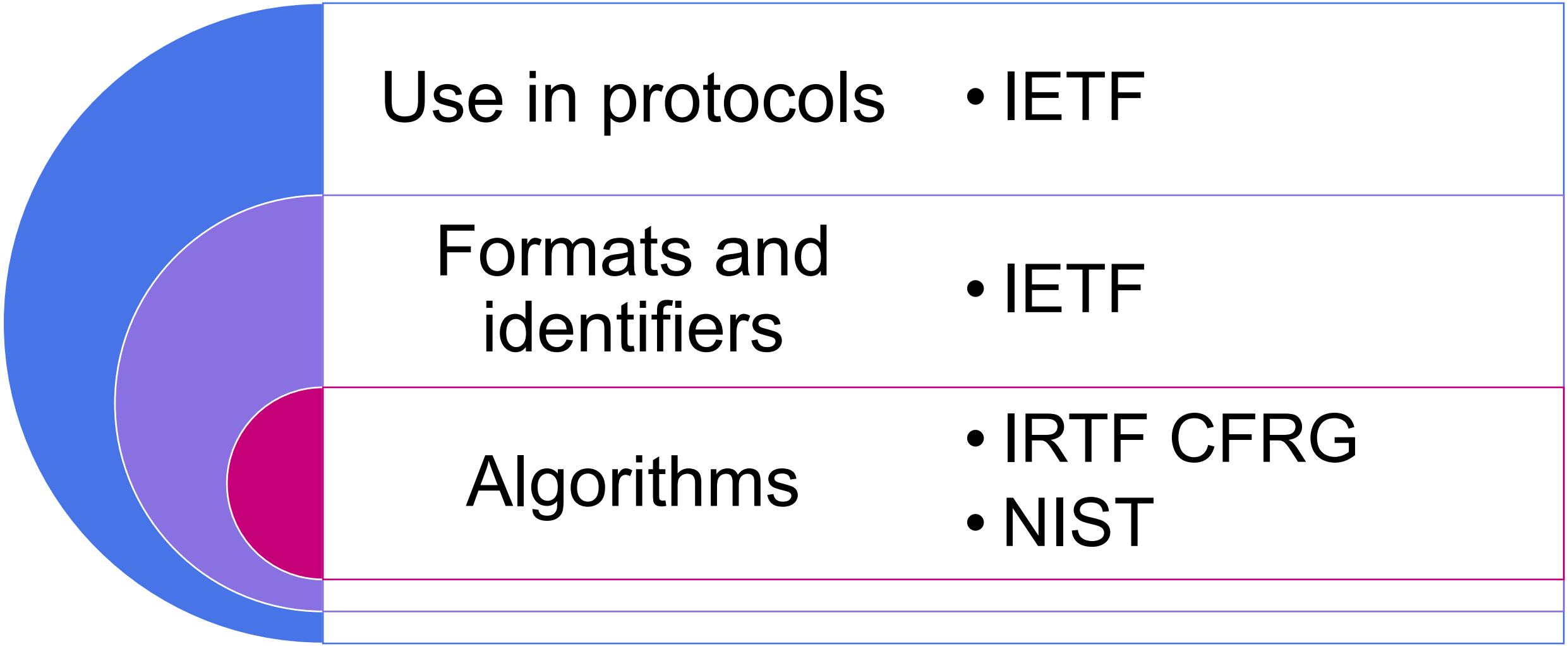
Douglas Stebila

https://www.douglas.stebila.ca/research/presentations/

NSERC CRSNG

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

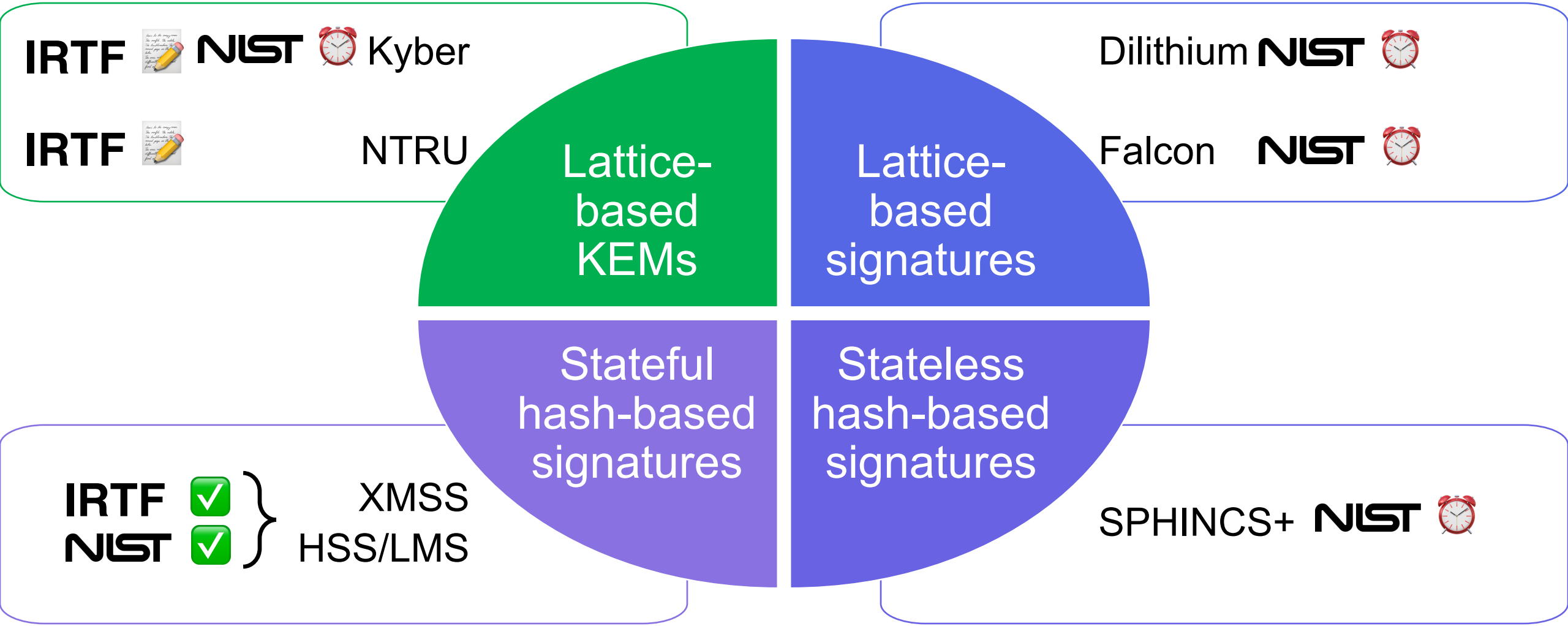
Levels of standardization



https://github.com/ietf-wg-pquip/state-of-protocols-and-pqc

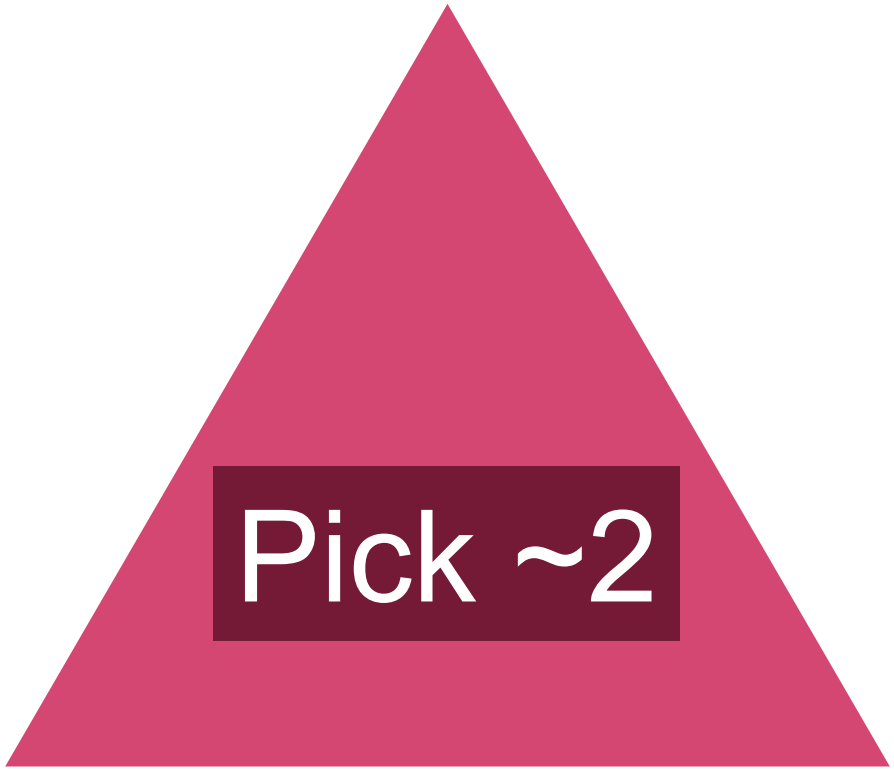
Standardizing PQ algorithms

PQ algorithms being standardized



Trade-offs with post-quantum crypto

Confidence in quantum-resistance



Fast computation

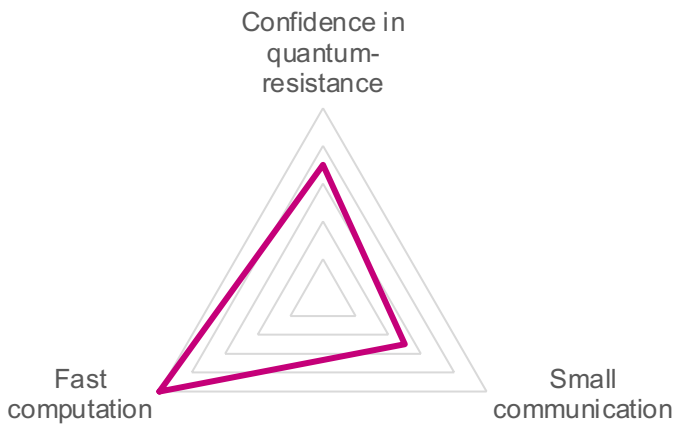
Small communication

Trade-offs with post-quantum crypto

RSA and elliptic curves

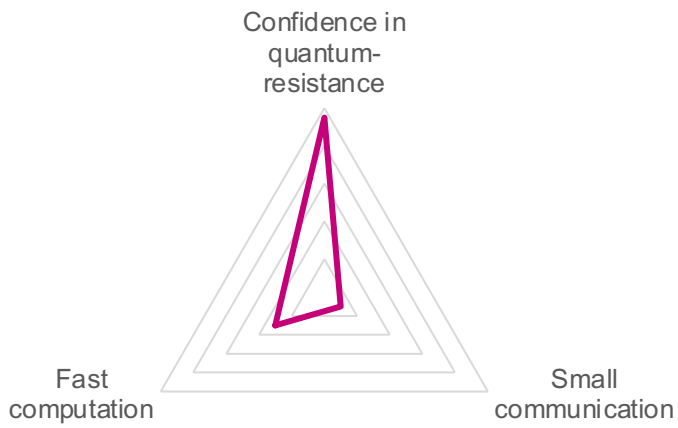
TLS handshake:
1.3 KB

Lattice-based cryptography



TLS handshake:
11.2 KB

Hash-based signatures



TLS handshake:
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 [1]

[1] Müller et al, ACM SIGCOMM Computer Communication Review 50(4):49–57, 2020.

Challenge: KEMs ≠ Diffie–Hellman

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

pk ⟶

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

⟵ ct

$k \leftarrow \mathsf{KEM.Decaps}(sk, ct)$

Responder's operation depends on sender's *pk* value

- 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

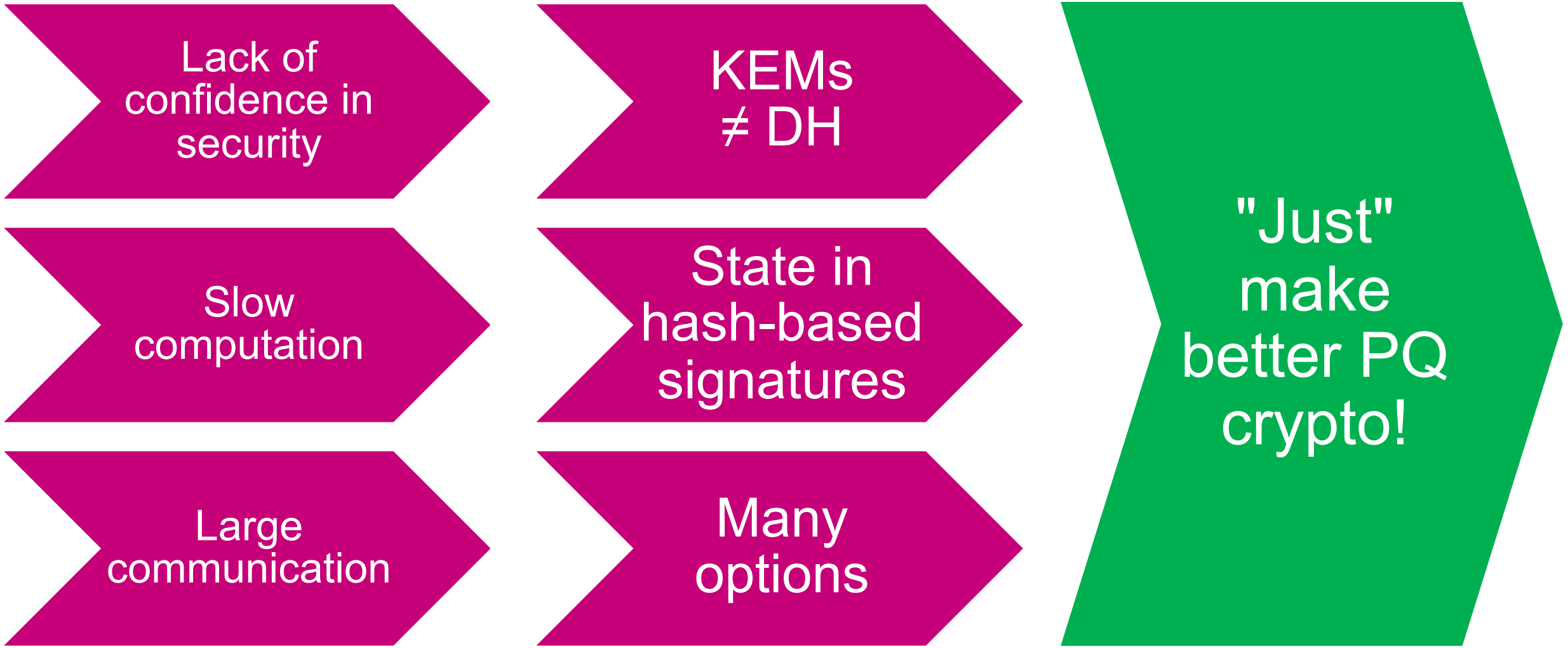
- Pure PQ?
- Hybrid?
  - With what elliptic curves?
  - With what combiner?

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

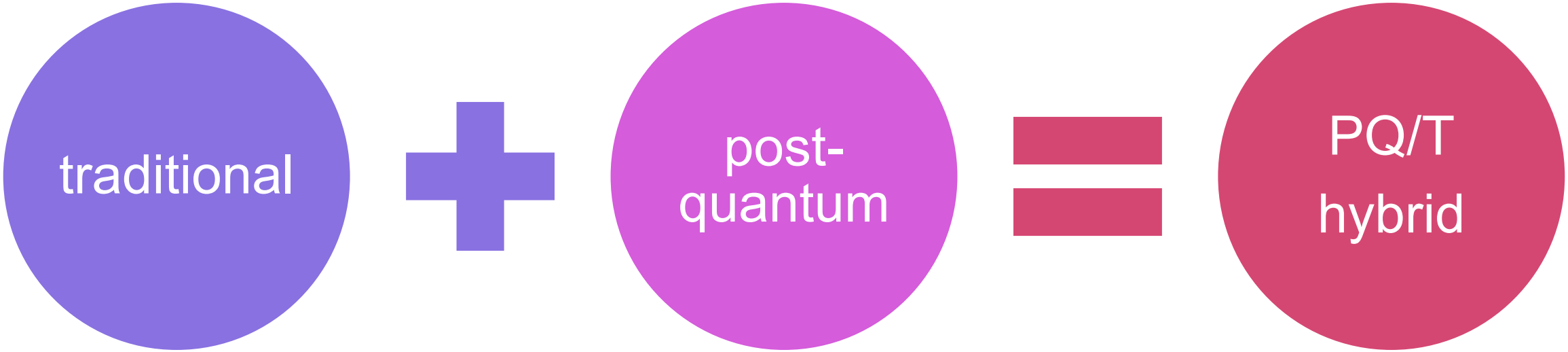


Addressing the challenges of using PQ crypto

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

Hybrid / composite

Hybrid approach: use traditional and post-quantum simultaneously such that successful attack needs to break both



Hybrid and composite terminology

	Composite hybrid	Non-composite hybrid
Protocol fields and messages	Unchanged	Changed to accommodate multiple elements
Cryptographic elements	Combined using a newly defined format	Unchanged

https://datatracker.ietf.org/doc/draft-driscoll-pqt-hybrid-terminology/

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)

Bottom half of slide from Mike Ounsworth presentation https://pkic.org/events/2023/post-quantum-cryptography-conference/pkic-pqcc-pqc-at-ietf-mike-ounsworth-entrust.pdf ; updated 2023-03-27 with corrected information from https://pkic.org/events/2023/post-quantum-cryptography-conference/pkic-pqcc-how-gc-preparing-for-pqc-melanie-anderson-jonathan-hammell-canadian-government.pdf

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

[1] https://datatracker.ietf.org/doc/draft-ounsworth-cfrg-kem-combiners/
[2] https://mailarchive.ietf.org/arch/msg/tls/F4SVeL2xbGPaPB2GW_GkBbD_a5M/

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

What is “post-quantum TLS”?

<table>
<tr><th rowspan="2">Pre-shared key (PSK) mode</th><th colspan="2">Key exchange</th></tr>
<tr><th>PQ-only</th><th>Hybrid</th></tr>
<tr><td rowspan="2">Already implemented
Still has key distribution problem
No forward secrecy</td><td colspan="2">Fairly easy to implement
Needed soonest: harvest now, decrypt later</td></tr>
<tr><td></td><td>Robust to 1 algorithm break
Possibly in demand during pre-certification</td></tr>
</table>

What is “post-quantum TLS”?

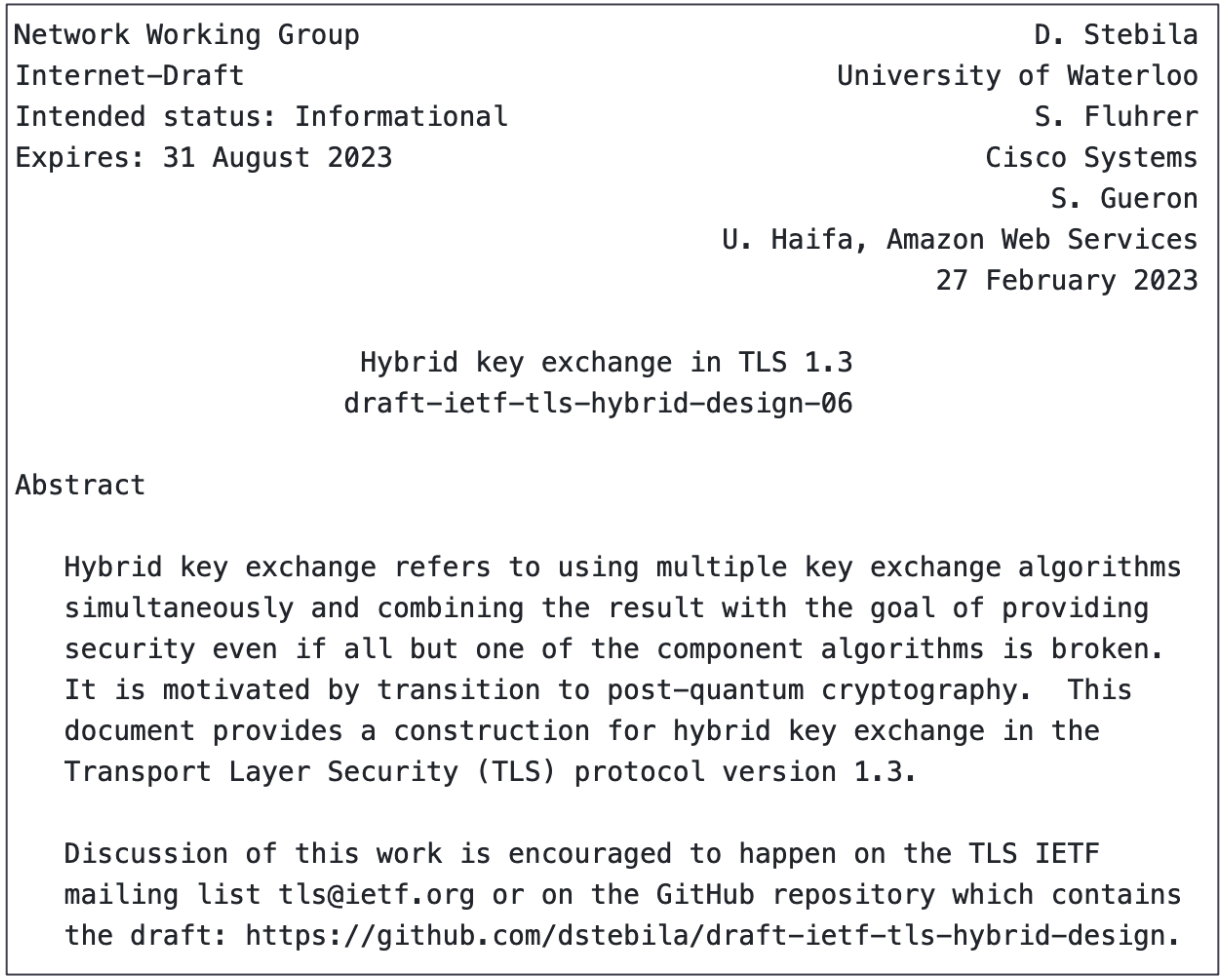
<table>
<tr><th rowspan="2">Pre-shared key (PSK) mode</th><th colspan="2">Key exchange</th><th colspan="2">Authentication</th></tr>
<tr><th>PQ-only</th><th>Hybrid</th><th>PQ-only</th><th>Hybrid / Composite</th></tr>
<tr><td rowspan="2">Already implemented
Still has key distribution problem
No forward secrecy</td><td colspan="2">Fairly easy to implement
Needed soonest: harvest now, decrypt later</td><td colspan="2">Requires coordination with certificate authorities
Less urgently needed: can't retroactively break authentication
Size ☹</td></tr>
<tr><td></td><td>Robust to 1 algorithm break
Possibly in demand during pre-certification</td><td></td><td>May not make sense in the context of a negotiated protocol like TLS</td></tr>
</table>

What is “post-quantum TLS”?

Pre-shared key (PSK) mode	Key exchange		Authentication		Alternative protocol designs
	PQ-only	Hybrid	PQ-only	Hybrid / Composite	
• Already implemented • Still has key distribution problem • No forward secrecy	• Fairly easy to implement • Needed sooner: harvest now, decrypt later		• Requires coordination with certificate authorities • Less urgently needed: can't retroactively break authentication • Size ☹		• e.g. 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	

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

https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/

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 [2] https://arxiv.org/abs/2211.14196

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]

[1] https://kemtls.org/ [2] https://datatracker.ietf.org/doc/draft-davidben-tls-merkle-tree-certs/
[3] https://arxiv.org/abs/2211.14196 [4] https://arxiv.org/abs/2302.05311

Wrapping up

Algorithm standardization status

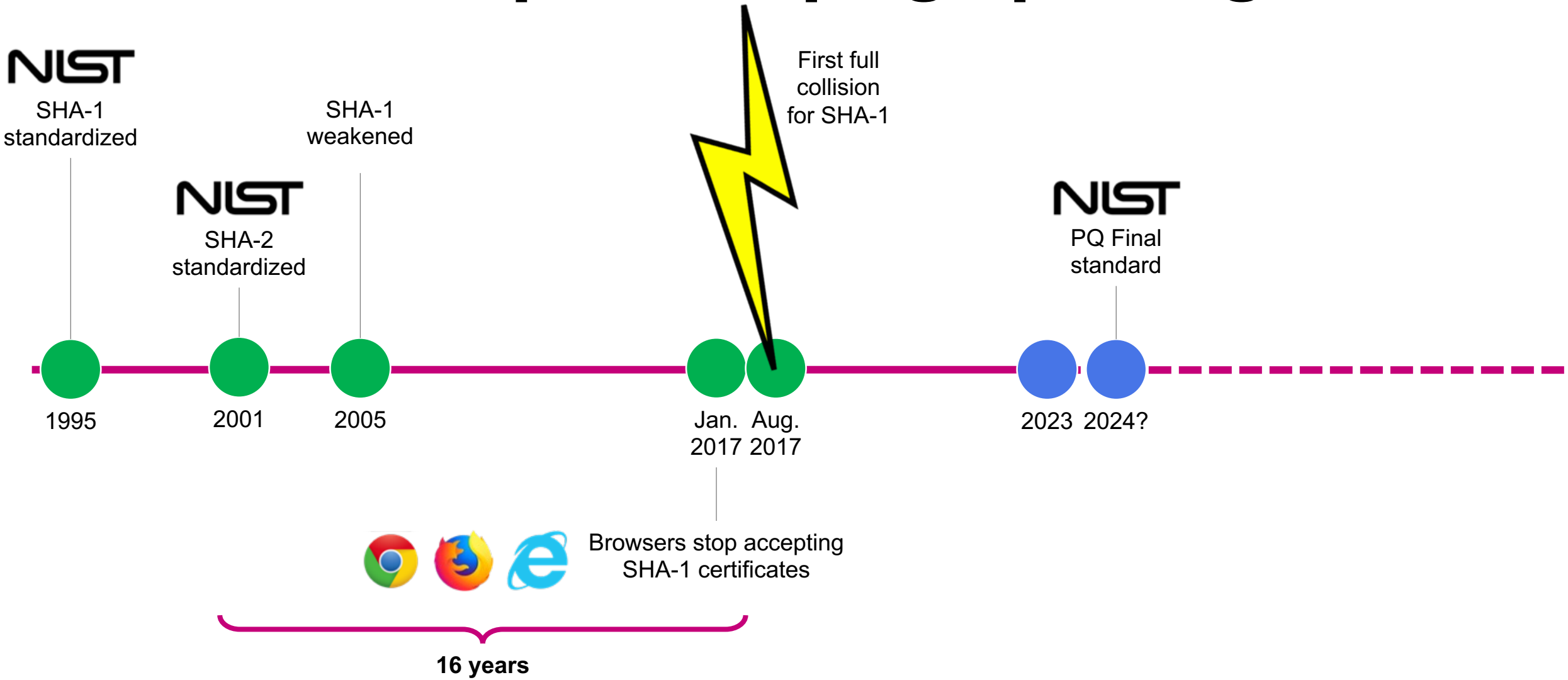
	Kyber	Dilithium	Falcon
Primary standardizer:	NIST	NIST	NIST
Status at NIST:	Draft standard pending	Draft standard pending	Draft standard pending
Status at IETF/IRTF:	CFRG draft available	No draft available	No draft available

	SPHINCS+	XMSS	LMS
Primary standardizer:	NIST	IRTF	IRTF
Status at NIST:	Draft standard pending	Approved in SP 800-208 (2020)	Approved in SP 800-208 (2020)
Status at IETF/IRTF:	No draft available	RFC 8391 (2018)	RFC 8554 (2019) Draft for new parameter sets

Protocol	Key exchange / PKE	Authentication	Alternatives
TLS 1.3 (secure channel)	Drafts: Hybrid Kyber	Prototypes	• AuthKEM / KEMTLS • TurboTLS • Merkle Tree certs.
X.509 (certificates)	Drafts: • Identifiers for Kyber	Drafts: • Identifiers and formats for Dilithium, LMS, XMSS, SPHINCS+ • Composite keys and signatures • Threshold composite • Binding non-composite certs	
Secure Shell (SSH) (secure channel)	Drafts: Hybrid Kyber OpenSSH: • Hybrid NTRU Prime	Prototypes	
IPsec (secure channel)	RFCs: PSK Drafts: hybrid, large messages	Drafts: • Hybrid non-composite • Negotiation	
CMS (secure email, …)	Drafts: KEMs, Kyber	RFCs: LMS Drafts: SPHINCS+	
DNSSEC (Domain Name Security)	Drafts: Stateful HBS		• Merkle Tree ladder • Request-based frag.
OpenPGP (secure email)	Drafts: • Composite Kyber	Drafts: • Composite Dilithium • PQ-only SPHINCS+	

https://github.com/ietf-wg-pquip/state-of-protocols-and-pqc

Timeline to replace cryptographic algorithms



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

Standardizing Post-Quantum Cryptography at the IETF

Douglas Stebila

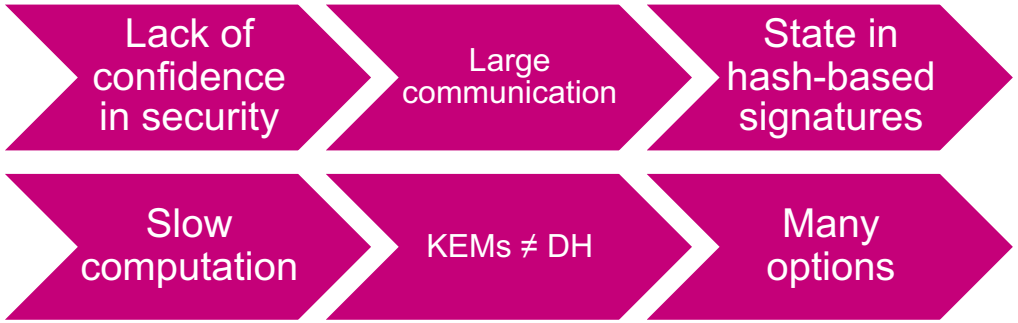
UNIVERSITY OF WATERLOO

https://www.douglas.stebila.ca/research/

Levels of standardization:

- Algorithms: NIST, IRTF CFRG
- Formats and identifiers: IETF
- Use in protocols: IETF

Trade-offs and challenges:

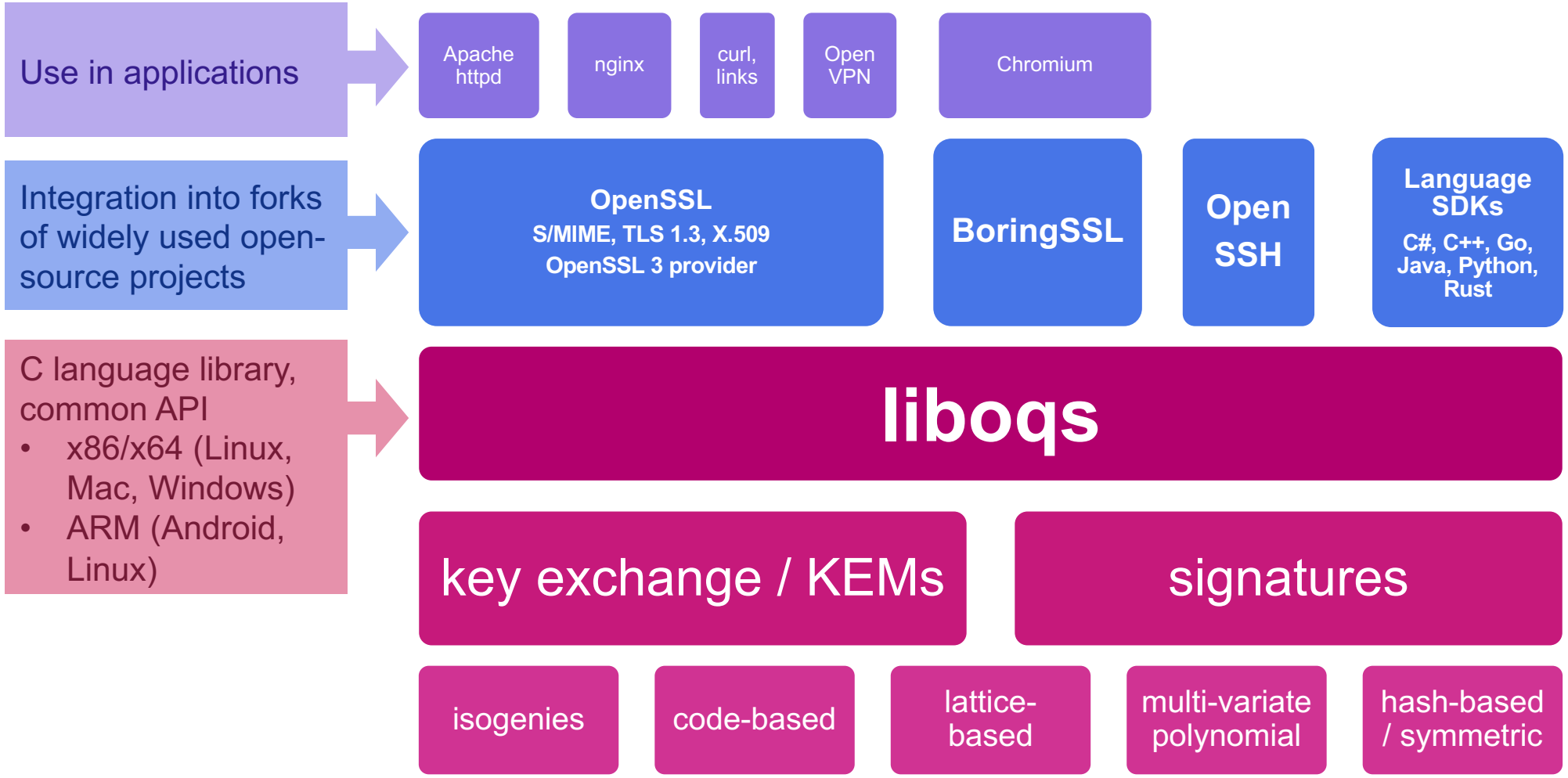


IETF/IRTF activities:

- **Algorithms:** Draft for Kyber, RFCs for XMSS & LMS
- **Protocols:**
  - TLS: Draft for hybrid KEX; several prototypes; alt. designs
  - X.509: Draft for algorithm identifiers, composite keys and signatures; IETF hackathon https://github.com/IETF-Hackathon/pqc-certificates
  - SSH: Draft for hybrid KEX; shipping in OpenSSH
  - IPsec, CMS, DNSSEC, OpenPGP
- **Working groups:**
  - TLS, LAMPS, …
  - PQUIP: Post-Quantum Use In Protocols https://github.com/ietf-wg-pquip/state-of-protocols-and-pqc

Appendix

Open Quantum Safe Project



Led by University of Waterloo

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

Additional contributors:
- Senetas
- PQClean project
- Individuals

Financial support:
- AWS
- Canadian Centre for Cyber Security
- Cisco
- NLNet
- NSERC
- Unitary Fund
- Verisign

https://openquantumsafe.org/ • https://github.com/open-quantum-safe/

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

TLS performance

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