Part 4b – Applications

Douglas Stebila McMaster University

SAC Summer School • Université d'Ottawa • August 14, 2017
https://www.douglas.stebila.ca/research/presentations

Funding acknowledgements:
NSERC CRSNG

Transitioning to PQ crypto

Retroactive decryption

- A passive adversary that records today's communication can decrypt once they get a quantum computer
  - Not a problem for some people
  - Is a problem for other people
- How to provide potential post-quantum security to early adopters?

Hybrid ciphersuites

- Use pre-quantum and post-quantum algorithms together
- Secure if either one remains unbroken

Need to consider backward compatibility for non-hybrid-aware systems

Why hybrid?

- Potential post-quantum security for early adopters
- Maintain compliance with older standards (e.g. FIPS)
- Reduce risk from uncertainty on PQ assumptions/parameters

Hybrid ciphersuites

	Key exchange	Digital signature
1	Hybrid traditional + PQ	Single traditional
2	Hybrid traditional + PQ	Hybrid traditional + PQ
3	Single PQ	Single traditional
4	Single PQ	Single PQ

Likely focus for next 10 years

Hybrid key exchange in TLS 1.2

Hybrid key exchange in TLS 1.2

Create a new DH-style ciphersuite with a new key exchange method

- Within the ClientKeyExchange and ServerKeyExchange, convey an ECDH public key and a PQ public key using some internal concatenation format
- Compute two shared secrets, use their concatenation as the premaster secret

Experiments for hybrid key exchange in TLS 1.2

Several papers and prototypes:

- Bos, Costello, Naehrig, Stebila, S&P 2015
- Bos, Costello, Ducas, Mironov, Naehrig, Nikolaenko, Raghunathan, Stebila, ACM CCS 2016
- Google Chrome experiment
- liboqs OpenSSL fork
  - https://openquantumsafe.org/

No backwards compatibility issues

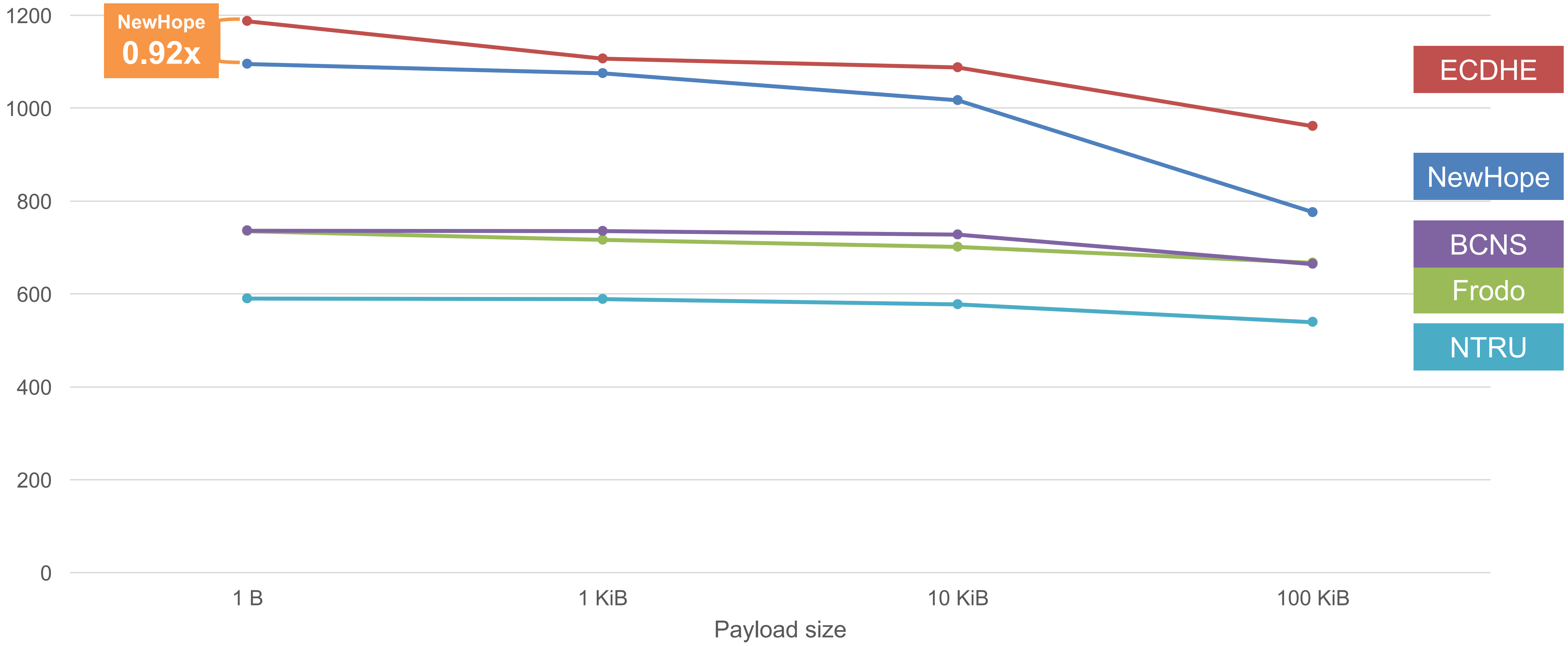
- https://www.imperialviolet.org/2016/11/28/cecpq1.html

https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

TLS connection throughput – hybrid w/ECDHE

ECDSA signatures

bigger (top) is better



x86_64, 2.6 GHz Intel Xeon E5 (Sandy Bridge) – server Google `n1-standard-4`, client `-32`

Note somewhat incomparable security levels

Security proofs for TLS 1.2

PRF-ODH

- Jager, Kohlar, Schage, Schwenk. Crypto 2012
- Krawczyk, Paterson, Wee. Crypto 2013

GapDH

- Kohlweiss, Maurer, Onete, Tackmann, Venturi. Indocrypt 2015

IND-CCA KEM

- Krawczyk, Paterson, Wee. Crypto 2013

Diffie–Hellman + computational randomness extractor

- Bhargavan, Fournet, Kohlweiss, Pironti, Strub, Zanella Béguelin. Crypto 2014

Post-quantum security of TLS 1.2

SIDH and LWE/ring-LWE are basically passively secure (IND-CPA) KEMs

Two approaches to provable active security in TLS 1.2:

1. Transform into IND-CCA KEM using e.g. Fujisaki–Okamoto transform then apply KPW13 proof

2. Move server signature later in the handshake so it authenticates the transcript, redo TLS 1.2 authentication proof to satisfy IND-CPA KEM / DDH + signature unforgeability
   - Approach taken in BCNS15/BCDNNRS16 proof (but not in experiments)
   - Note proof only against a classical adversary

Hybrid key exchange in TLS 1.3

Hybrid key exchange in TLS 1.3

Three possible techniques:

Technique 1. Naïve:

- Define new named groups for each hybrid key exchange combination, with semantics internally defined by the named group
- Simplest; requires no changes to TLS 1.3
- Combinatorial explosion of ciphersuites
- Theoretically no backwards compatibility issues with non-aware TLS 1.3 implementations

Hybrid key exchange in TLS 1.3

Technique 2. draft-whyte-qsh-tls13-04:

- Define new generic named groups for hybrid key exchanges, with a mapping (in a new extension) from the generic named groups to the actual hybrid named groups they comprise and semantics for parsing KeyShares containing hybrid keys
- Supports up to 10 hybrid algorithms in a single key exchange
- Requires adding new extension, plus logic for handling hybrid named groups and hybrid keyshares; hybrid named groups have no external meaning
- Theoretically no backwards compatibility issues with non-aware TLS 1.3 implementations

[Whyte, Zhang, Fluhrer, Garcia-Morchon, March 2017]

Hybrid key exchange in TLS 1.3

Technique 3. draft-schanck-tls-additional-keyshare-00

- Add second extension for conveying additional KeyShare using same data structures as existing KeyShare data structure
- Supports up to 2 hybrid algorithms in a single key exchange (though approach is extensible)
- Requires adding new extension, plus logic for handling additional extension and key schedule updates
- Theoretically no backwards compatibility issues with non-aware TLS 1.3 implementations

[Schanck, Stebila, April 2017]

Security proofs for TLS 1.3

DDH
- OPTLS, 1-RTT mode [Krawczyk, Wee. EuroS&P 2016]

GapDH standard model
- OPTLS, 1-RTT semi-static mode [KW16]
- OPTLS, 1-RTT semi-statis early data mode [KW16]
- Draft 10 [Li, Xu, Zhang, Feng, Hu. S&P 2016]
- Draft ?? [Kohlweiss, Maurer, Onete, Tackmann, Venturi. Indocrypt 2015]

GapDH random oracle model
- Draft 18 [Bhargavan, Blanchet, Kobeissi. S&P 2017]

PRF-ODH
- Main handshake, draft 5, 10 [Dowling, Fischlin, Günther, Stebila. ACM CCS 2015, eprint]
- 0-RTT, draft 12 [Fischlin, Günther. EuroS&P 2017]

Symbolic
- Draft 10 [Cremers, Horvat, Scott, van der Merwe. S&P 2016]

Post-quantum security of TLS 1.3

- **Cannot use GapDH proofs** for LWE/ring-LWE since it does not satisfy GapDH due to search-decision equivalence
- **Cannot use PRF-ODH proofs** for LWE/ring-LWE due to key reuse attacks
  - Possible workaround: some PRF-ODH proofs use a very small number of reuses (e.g., 2), whereas attacks use many more (e.g., ≥ 500), but no results on when this is safe

Post-quantum security of TLS 1.3

- Could **transform** post-quantum KEMs from IND-CPA to IND-CCA using FO transform
  - May need to have different parameters due to correctness probability
- Or **directly construct** IND-CCA KEMs
  - [Albrecht, Orsini, Paterson, Peer, Smart, Eprint 2017]

- But either case needs **new TLS 1.3 proofs** that generically use an **IND-CCA KEM** à la [KPW13]
- (Also need to upgrade proofs to quantum adversary and quantum random oracle model.)

Hybrid authentication

Hybrid authentication in TLS 1.3

Need to negotiate traditional + PQ algorithms

Need to convey

1. Traditional subject public key
2. Traditional CA signature and chain
3. PQ subject public key
4. PQ CA signature and chain

Security issues for hybrid authentication

- Should the PQ CA signature cover both the traditional and PQ components?
- Should the traditional CA signature cover both the traditional and PQ components?
- Neither is necessarily possible due to backwards-compatibility issues
- => Is it bad if an adversary can separate out one signature scheme from the certificate?
- Some discussion of these issues in [Bindel, Herath, McKague, Stebila, PQCrypto 2017]

Protocol design issues for hybrid authentication

- How to convey second subject public key, CA signature, and chain?

- As a monolithic hybrid signature scheme?
- As a second certificate in a TLS extension?
  - Client auth: TLS 1.3 post-handshake client authentication might work
  - Server auth: No clear mechanism in TLS 1.3 directly; maybe draft-sullivan-tls-exported-authenticator?
- In a TLS 1.3 Certificate extension?
  - Still need to convey second signature?
- As an extension in the traditional certificate?
  - Need standardized semantics for both PKI and TLS
  - See [Brown et al. ICMC 2017] or [Bindel, Herath, McKague, Stebila PQCrypto 2017]

Hybrid signatures in X.509 certificates

- How to convey multiple public keys in a single certificate?
- How to sign a single certificate with multiple CA algorithms?

- **X.509 extensions**
  - Can carry arbitrary additional data
  - Put a second "post-quantum" certificate as an extension inside a traditional (RSA/ECDSA) certificate
  - Post-quantum aware software recognizes both and processes both
  - Old software ignores "non-critical" extensions
    - => backwards compatible

Compatibility of large extensions in certs in TLS

	Extension size in KiB				
	1.5	3.5	9.0	43.0	1333.0
Libraries (library's command-line client talking to library's command-line server)					
GnuTLS 3.5.11	✓	✓	✓	✓	×
Java SE 1.8.0_131	✓	✓	✓	✓	✓
mbedTLS 2.4.2	✓	✓	✓	×	×
NSS 3.29.1	✓	✓	✓	✓	×
OpenSSL 1.0.2k	✓	✓	✓	✓	×
Web browsers (talking to OpenSSL's command-line server)					
Apple Safari 10.1 (12603.1.30.0.34)	✓	✓	✓	✓	✓
Google Chrome 58.0.3029.81	✓	✓	✓	✓	×
Microsoft Edge 38.14393.1066.0	✓	✓	✓	×	×
Microsoft IE 11.1066.14393.0	✓	✓	✓	×	×
Mozilla Firefox 53.0	✓	✓	✓	✓	×
Opera 44.0.2510.1218	✓	✓	✓	✓	×

[Bindel, Herath, McKague, Stebila, PQCrypto 2017]

Hybrid signatures in S/MIME encrypted email

- How to convey multiple signatures on a single message?

- S/MIME data structures allow multiple parallel signatures
  - But most software tries to validate **all** parallel signatures and rejects if any of them fail
  - => Not backwards compatible

- Various options with extension fields (attributes)

Research in hybrid cryptography

- For each type of primitive (key exchange, public key encryption, digital signatures), what possible ways can we combine algorithms?
  - $s_1 = \mathrm{Sign}_1(sk_1, m)$; $s_2 = \mathrm{Sign}_2(sk_2, m)$; $sig = (s_1, s_2)$
  - $s_1 = \mathrm{Sign}_1(sk_1, m)$; $s_2 = \mathrm{Sign}_2(sk_2, s_2)$; $sig = (s_1, s_2)$
  - $s_1 = \mathrm{Sign}_1(sk_1, m)$; $s_2 = \mathrm{Sign}_2(sk_2, m \,||\, s_1)$; $sig = (s_1, s_2)$

- Are these schemes secure against quantum adversaries?
- How quantum is the adversary?
  - Classical adversary now, quantum later
  - Quantum adversary with **only classical** access to signing/decryption oracles
  - Quantum adversary with quantum access to **random oracle**
  - Quantum adversary with quantum access to **signing/decryption oracles**

Open Quantum Safe

https://openquantumsafe.org/

Open Quantum Safe

- MIT-licensed open-source project on Github
  - https://openquantumsafe.org/
  - https://github.com/open-quantum-safe/
- liboqs: C language library, common API

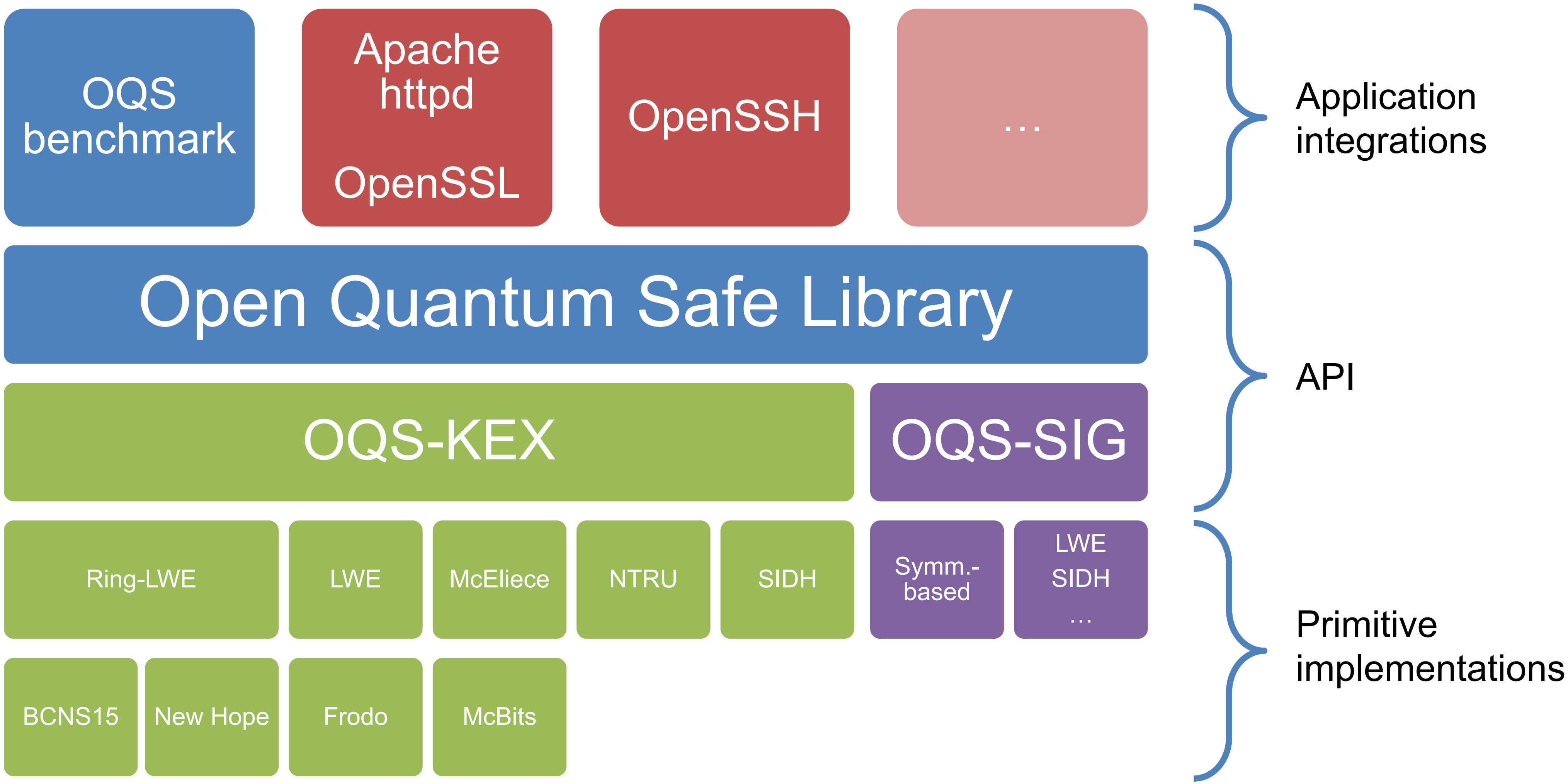
Open Quantum Safe

1. Collect post-quantum implementations together
   - Our own software
   - Thin wrappers around existing open source implementations
   - Contributions from others

2. Enable direct comparison of implementations
   - See also eBACS/SUPERCOP

3. Support prototype integration into application level protocols
   - Don’t need to re-do integration for each new primitive – how we did Frodo experiments

Open Quantum Safe architecture



liboqs: Current algorithms

Key exchange

- **Ring-LWE**:
  - BCNS15
  - NewHope
  - MSR NewHope improvements
- **LWE**: Frodo
- **M-LWE**: Kyber
- **NTRU**
- **SIDH (Supersingular isogeny Diffie–Hellman)**:
  - MSR
  - IQC
- **Code**: McBits

Digital signatures

- **Symmetric-based**:
  - Picnic

liboqs: Benchmarking

- Built-in key exchange benchmarking suite
  - `./test_kex --bench`
- Gives cycle counts and ms runtimes

liboqs: Application integrations

OpenSSL v1.0.2:

- Ciphersuites using key exchange algorithms from liboqs
- Integrated into `openssl speed` benchmarking command and `s_client` and `s_server` command-line programs

- Track OpenSSL 1.0.2 stable with regular updates
  - https://github.com/open-quantum-safe/openssl

- Successfully used in Apache httpd and OpenVPN (with no modifications!)

OpenSSH:

- Using key exchange algorithms from liboqs
- Patch contributed by Microsoft Research
  - https://github.com/Microsoft/PQCrypto-PatchforOpenSSH

OQC contributors and acknowledgements

Project leaders

- Michele Mosca and Douglas Stebila

Planning & discussions

- Scott Vanstone and Sherry Shannon Vanstone (Trustpoint)
- Matthew Campagna (Amazon Web Services)
- Alfred Menezes, Ian Goldberg, and Guang Gong (University of Waterloo)
- William Whyte and Zhenfei Zhang (Security Innovation)
- Jennifer Fernick, David Jao, and John Schanck (University of Waterloo)

Software contributors

- Mike Bender
- Tancrède Lepoint (SRI)
- Shravan Mishra (IQC)
- Christian Paquin (MSR)
- Alex Parent (IQC)
- Douglas Stebila (McMaster)
- Sebastian Verschoor (IQC)

+ Existing open-source code

Getting involved and using OQS

https://openquantumsafe.org/

If you're writing post-quantum implementations:

- We'd love to coordinate on API
- And include your software if you agree

If you want to prototype or evaluate post-quantum algorithms in applications:

- Maybe OQS will be helpful to you

We'd love help with:

- Code review and static analysis
- Signature scheme implementations
- Additional application-level integrations

Summary

Quantum-safe crypto

Classical post-quantum crypto

Hash-based
- Merkle signatures
- Sphincs

Code-based
- McEliece
- Niederreiter

Multivariate
- multivariate quadratic

Lattice-based
- NTRU
- learning with errors
- ring-LWE

Isogenies
- supersingular elliptic curve isogenies

Quantum crypto

Quantum key distribution

Quantum random number generators

Quantum channels

Quantum blind computation

NIST Post-quantum Crypto Project timeline

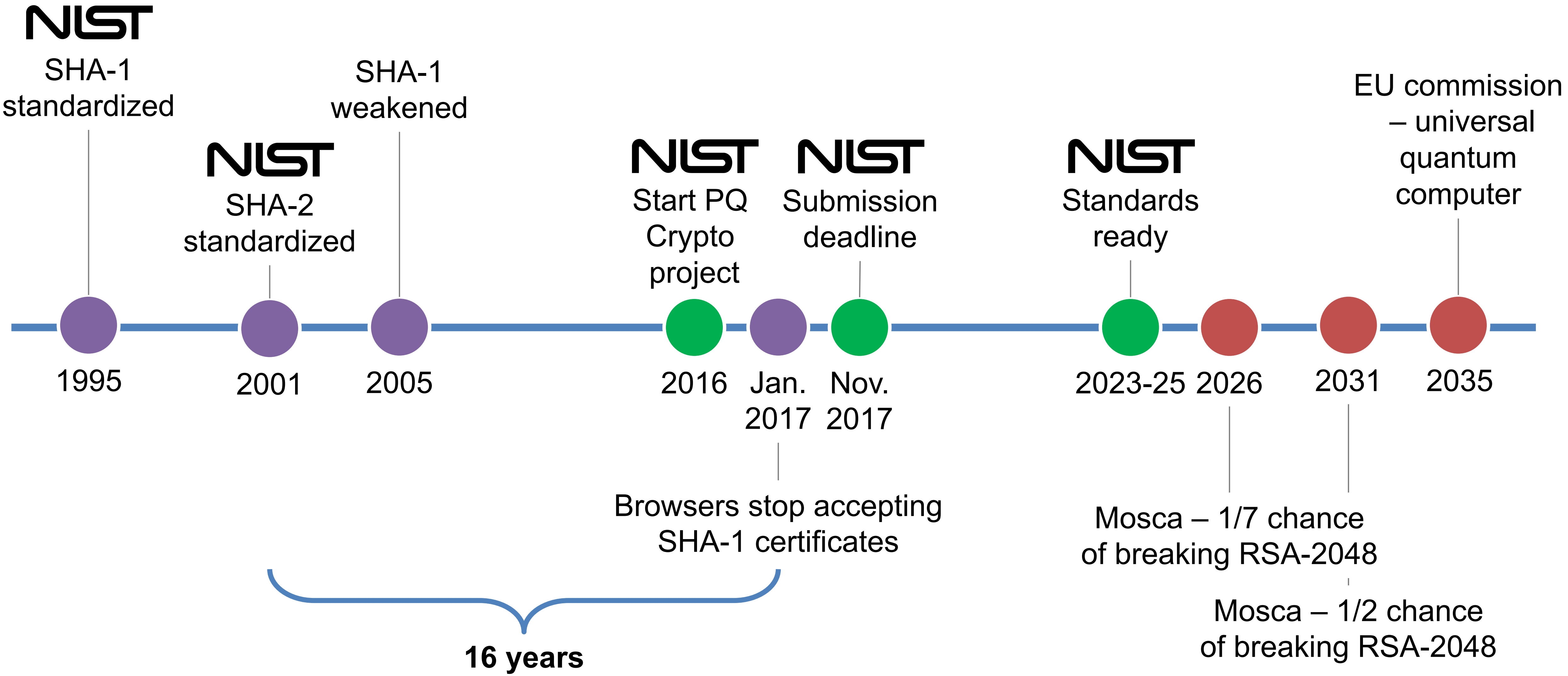
http://www.nist.gov/pqcrypto

December 2016	Formal call for proposals
November 2017	**Deadline for submissions**
3-5 years	Analysis phase
2 years later (2023-2025)	Draft standards ready

"**Our intention is to select a couple of options** for more immediate standardization, as well as to eliminate some submissions as unsuitable. … The goal of the process is **not primarily to pick a winner**, but to document the strengths and weaknesses of the different options, and to analyze the possible tradeoffs among them."

http://csrc.nist.gov/groups/ST/post-quantum-crypto/faq.html#Q7

Timeline


NIST
SHA-1 standardized
1995
NIST
SHA-2 standardized
2001
SHA-1 weakened
2005
NIST
Start PQ Crypto project
2016
Jan. 2017
Browsers stop accepting SHA-1 certificates
NIST
Submission deadline
Nov. 2017
NIST
Standards ready
2023-25
2026
Mosca – 1/7 chance of breaking RSA-2048
2031
Mosca – 1/2 chance of breaking RSA-2048
EU commission – universal quantum computer
2035
16 years