Preparing for post-quantum cryptography in TLS

Douglas Stebila McMaster University

Funding acknowledgements:

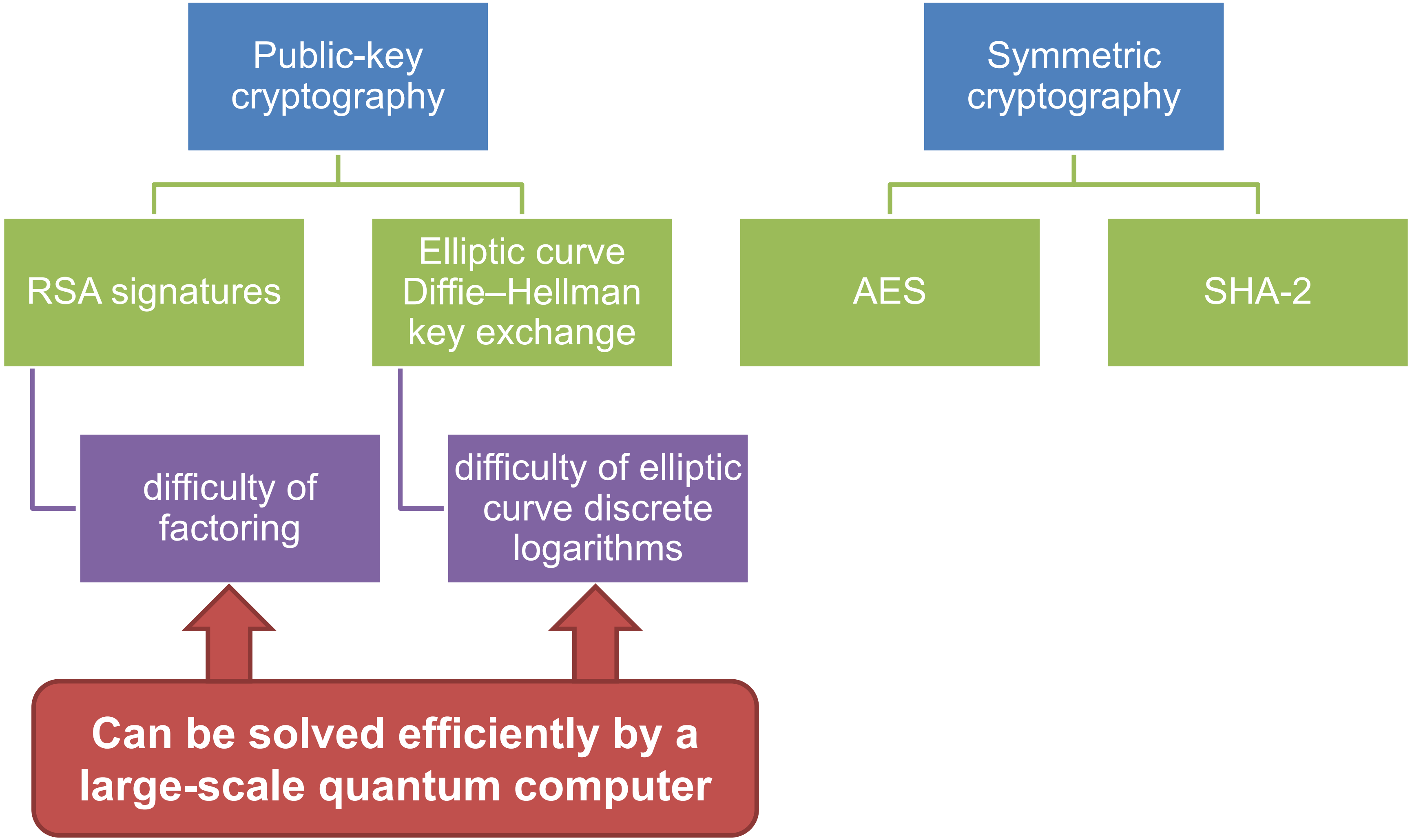
NSERC CRSNG

TLS:DIV workshop • April 30, 2017

Motivation

Contemporary cryptography

`TLS-ECDHE-RSA-AES128-GCM-SHA256`



When will a large-scale quantum computer be built?

"I estimate a 1/7 chance of breaking RSA-2048 by 2026 and a 1/2 chance by 2031."

— Michele Mosca, November 2015
https://eprint.iacr.org/2015/1075

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

Post-quantum / quantum-safe crypto

No known exponential quantum speedup

Hash-based

- Merkle signatures
- Sphincs

Code-based

- McEliece

Multivariate

- multivariate quadratic

Lattice-based

- NTRU
- learning with errors (LWE)
- ring-LWE

Isogenies

- supersingular elliptic curve isogenies (SIDH)

Post-quantum signature sizes

	Public key		Signature	
RSA 3072-bit	Small	0.3 KiB	Small	0.3 KiB
ECDSA `nistp256`	Very small	0.03 KiB	Very small	0.03 KiB
Hash-based (stateful)	Small	0.9 KiB	Medium	3.6 KiB
Hash-based (stateless)	Small	1 KiB	Large	32 KiB
Lattice-based (ignoring tightness)	Medium	1.5–8 KiB	Medium	3–9 KiB
Lattice-based (respecting tightness)	Very large	1330 KiB	Small	1.2 KiB
SIDH	Small	1.5 KiB	Very large	704 KiB

See [Bindel, Herath, McKague, Stebila PQCrypto 2017] for details

Post-quantum key exchange performance

	Speed		Communication	
RSA 3072-bit	Fast	4 ms	Small	0.3 KiB
ECDH `nistp256`	Very fast	0.7 ms	Very small	0.03 KiB
Code-based	Very fast	0.5 ms	Very large	360 KiB
NTRU	Very fast	0.3–1.2 ms	Medium	1 KiB
Ring-LWE	Very fast	0.2–1.5 ms	Medium	2–4 KiB
LWE	Fast	1.4 ms	Large	11 KiB
SIDH	Slow	35–400 ms	Small	0.5 KiB

See [Bos, Costello, Ducas, Mironov, Naehrig, Nikolaenko, Raghunathan, Stebila, ACM CCS 2016] for details/methodology

Assumptions for post-quantum KEMs – Supersingular elliptic curve isogenies (SIDH)

Supersingular computational Diffie–Hellman problem (SSCDH)

- Given public keys
  $pk_1=f(param, sk_1)$,
  $pk_2=f(param, sk_2)$,
  find
  $ssk=g(pk_2, sk_1)=g(pk_1, sk_2)$

Like CDH

Supersingular decisional Diffie–Hellman problem (SSDDH)

- Given public keys
  $pk_1=f(param, sk_1)$,
  $pk_2=f(param, sk_2)$,
  distinguish
  $ssk=g(pk_2, sk_1)=g(pk_1, sk_2)$
  from
  ssk_{rand}

Like DDH (IND-CPA KEM)

[De Feo, Jao, Plût, PQCrypto 2011]

Assumptions for post-quantum KEMs – Learning with errors, ring-LWE

Search LWE:

- Given public key pk=(A, b=f(A, sk, rand)), find sk.

Like discrete log

Decision LWE:

- Distinguish
  pk=(A, b=f(A, sk, rand))
  from
  pk_{rand}=(A, rand).

Partially commutative PRF?

[Regev, STOC 2005]

Assumptions for post-quantum KEMs – Learning with errors, ring-LWE

Lindner–Peikert KEM/PKE:

- Given public keys
  $pk_1=(A, b_1=f(A, sk_1, rand_1))$
  $pk_2=(A, b_2=f(A, sk_2, rand_2))$
  distinguish
  $ssk=g(b_2, sk_1)=g'(b_1, sk_2)$
  from
  ssk_{rand}

Like DDH
(IND-CPA KEM)

Follows from
decision LWE

[Lindner, Peikert, CT-RSA 2011]

Assumptions for post-quantum KEMs – Learning with errors, ring-LWE

- Search easy => decision easy
  - Straightforward reduction (DLOG easy => DDH easy)

- Decision easy => search easy
  - "Search-decision equivalence" [Regev, STOC 2005]

Public key validation

- **No public key validation possible** in IND-CPA KEMs from LWE/ring-LWE and SIDH
- **Key reuse in LWE/ring-LWE** leads to real attacks following from search-decision equivalence
  - Comment in [Peikert, PQCrypto 2014]
  - Attack described in [Fluhrer, Eprint 2016]

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

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

- 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

Hybrid key exchange in TLS 1.3

Technique 3. draft-schanck-tls-additional-keyshare-00
[Schanck, Stebila, April 2017]:

- 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

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 in TLS 1.3

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?

1. As a monolithic hybrid signature scheme?
2. 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?
3. In a TLS 1.3 Certificate extension?
   - Still need to convey second signature?
4. 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]

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]

Summary

Preparing for post-quantum cryptography in TLS

Douglas Stebila
McMaster University
https://www.cas.mcmaster.ca/~stebilad/

TLS 1.3 experimentation:

- Need information on key size / signature size limits for compatibility

TLS 1.3 protocol design:

- Need places to put secondary key exchange in handshake and key schedule
- Need places to put secondary server authentication
- May need to handle larger-than-desirable objects
- May have multiple options with various tradeoffs
  near/at end of NIST PQ project => no clear single winner

TLS 1.3 security analysis:

- Need proofs using generic IND-CCA KEM
  - And quantum adversary / quantum random oracle model
- Security models and proofs for hybrids
- Check symmetric primitives too (Kaplan et al. Crypto 2016)

Similar issues for Signal, QUIC, …

Open Quantum Safe project
https://openquantumsafe.org/

- Open-source C library with multiple PQ key exchange algorithms (PQ signatures soon)
- TLS 1.2 prototype in OpenSSL
- TLS 1.3 prototype later this year