

Post-Quantum Cryptography

CS7NS5/CSU44032

<https://down.dsg.cs.tcd.ie/cs7053/>
stephen.farrell@cs.tcd.ie

Top Level Message

- Don't panic, most people don't need to care just yet (but getting closer)
- Cryptographers are studying new but fairly well-defined algorithms that aim to provide good security even in the face of a cryptographically relevant quantum computer (CRQC)
- Engineers are starting to consider how to integrate those algorithms into protocols and applications
- Could well be that hybrid cryptographic solutions (using both well-understood “classic” algorithms and new, less well-understood, post-quantum algorithms) will be deployable in the next few years

Resources

- We don't (yet!) need to understand all the details here but to delve another layer deeper...
- Ericsson: "Quantum-Resistant Cryptography", John Mattsson et al (40pp)
 - <https://arxiv.org/abs/2112.00399>
 - Summary in Ericsson Technology Review: "Quantum technology and its impact on security in mobile networks"
 - <https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/ensuring-security-in-mobile-networks-post-quantum>
- ARM: "Whitepaper: Post-Quantum Cryptography", Hanno Becker
 - <https://community.arm.com/arm-research/m/resources/1002>
- ICANN-sponsored backgrounder by Hilarie Orman (a very good security/crypto person and a really excellent writer!)
 - <https://eprint.iacr.org/2021/1637.pdf> ← I'd start with this one
- Interesting looking diagram with sources (not sure myself how much weight I'd put on it)
 - https://sam-jaques.appspot.com/quantum_landscape_2022

Overview

- The quantum computing problem for cryptography
- Cryptographically relevant quantum computer (CRPQ)
- Effects on “classic” cryptography
- When does this matter?
- NIST PQ Competition Algorithms
- Hash-based (Stateful) signatures
- Hybrid application/protocol approaches
- Quantum Key Distribution

Quantum Computing

- Classical computers (today's) based on binary bits; Quantum computers based on (coherent) qubits that allow running algorithms that explore many different values at once
- Many, many well-resourced people are working to develop quantum computers that could improve on current classic computers
 - There may be many other uses for quantum computers, they're not only for breaking crypto
- Cryptographically relevant quantum computers (CRQCs) do not exist, and might never
 - Or they might... someday, we don't know
- At present quantum computers seem unlikely to be general purpose computers, seems more like each hardware device will be built to solve a particular problem
- Current experiments only support tiny numbers of physical qubits (10's up to ~100)
- Error-correction/de-coherence problem means many physical qubits needed for each logical qubit
- Likely millions of qubits needed to be cryptographically relevant

The quantum problem

- No real impact on hash functions
- Grover's algorithm is a more efficient brute-force attack/search meaning attacks on 128-bit keys take 2^{64} operations (not speedy ops nor easy to parallelise so far)
 - Easy solution, if even needed: use 256 bit symmetric keys (AES-256/ChaCha20)
- **Shor's algorithm** describes how to quickly factor integers and hence break RSA
 - Variants solve discrete log problem, breaking (EC)DH
 - Hard solution: new post-quantum (PQ) algorithms
- Beware though: attacks only get worse – new quantum attacks may well be discovered in future (even before a cryptographically relevant quantum computer exists)

When is this a problem?

- Signals intelligence agencies seem to like to store all ciphertext, as far as practical, so we should assume all ciphertexts/cryptographic sessions are stored somewhere
 - Don't forget long-lived meta-data: phone numbers, email addresses, usernames
- If a cryptographically relevant quantum computer were to exist in 20xx then data that is protected today using classic public key algorithms, and that will still be sensitive in 20xx, is at risk now
 - Data that will not be sensitive in 20xx is unaffected
- So: analyse what data you're dealing with that will be sensitive in 10, 20, 30 years time and keep an eye on developments
 - In a very few years time, there will likely be usable hybrid approaches to protecting such data (more later)

NIST Competition

- Started in 2017, first “winners” in 2022, appears to have attracted best talent for entries
 - <https://csrc.nist.gov/projects/post-quantum-cryptography>
- Looking for Key Encapsulation Mechanisms (KEMs) and Signature algs
 - KEMs more pressing – alternative to RSA key transport or DH key establishment
 - Signatures less urgent and we have stateful sigs already (later)
- After 3 elimination rounds, some winners announced in July 2022
 - KEM: **Kyber**
 - Signature: **Dilithium**, Falcon, SPHINCS+
- Nitty parameter details for the above still need to be determined and the algorithms still need to go through NIST’s formal standardisation process before they’re done (in maybe 2024)

NIST Competition: winner all right?

- As well as announcing winners NIST also announced a “round 4” of evaluation for KEMs and issued a new call for submissions for signatures
- BUT...
 - Kyber IPR situation wasn't quite clean so NIST agreed/paid-for licenses
<https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/selected-algos-2022/nist-pqc-license-summary-and-excerpts.pdf>
 - Not sure if that'll be enough, time will tell
 - NTRUprime was not selected but proponents and some deployments continue working on that
 - Selected signature schemes may create an API issue for large to-be-signed values
 - One of the round 4 KEM algorithms (SIKE) was busted in August 2022
<https://www.schneier.com/blog/archives/2022/08/sike-broken.html>
 - One of the round 3 signature finalists (Rainbow) had been busted in Feb 2022
<https://eprint.iacr.org/2022/214>

BUSTED



KEM algorithm	Generate key	Encaps.	Decaps.	Public key size	Encaps. size
NTRU (ntruhs2048509)	0.048 ms	0.0073 ms	0.012 ms	699 B	699 B
Kyber (kyber512)	0.0070 ms	0.011 ms	0.0084 ms	800 B	768 B
SABER (lightsaber2)	0.012 ms	0.016 ms	0.016 ms	672 B	736 B
Classic McEliece (mceliece348864)	14 ms	0.011 ms	0.036 ms	261120 B	128 B
SIKE (SIKEp434_compressed)	3.0 ms	4.4 ms	3.3 ms	197 B	236 B
ECDH (X25519) (non-PQC)	0.038 ms	0.044 ms	0.044 ms	32 B	32 B
ECDH (P-256) (non-PQC)	0.074 ms	0.18 ms	0.18 ms	32-64 B	32-64 B
RSA-3072 (non-PQC)	400 ms	0.027 ms	2.6 ms	384 B	384 B

Table 1: Single-core median performance on Intel Xeon E-2124 3.3 GHz of some of the NIST PQC KEM algorithm candidates (and some current non-PQC alternatives) at NIST PQC security level 1. Source: r24000, supercop-20210604 at [107]. The measurements for RSA-3072 are estimated by taking the measurements for verify and sign from Table 1 as encapsulate and decapsulate, respectively. The measurements for SIKE are for a different platform: Intel Core i7-6700 3.4 GHz [108].

BUSTED →

Signature algorithm	Generate key	Sign	Verify	Public key size	Signature size
Falcon (falcon512dyn)	5.9 ms	0.23 ms	0.029 ms	897 B	666 B
Dilithium (dilithium2aes)	0.015 ms	0.041 ms	0.019 ms	1312 B	2420 B
Rainbow (rainbow1aclassic363232)	2.7 ms	0.017 ms	0.0087 ms	161600 B	64 B
SPHINCS+ (SPHINCS+-SHA-256-128s-simple)	27 ms	210 ms	0.28 ms	32 B	7856 B
LMS (using SHA-256, limited to 2^{20} messages)	-	-	-	56 B	2828 B
Ed25519 (non-PQC)	0.014 ms	0.015 ms	0.050 ms	32 B	64 B
ECDSA (P-256) (non-PQC)	0.029 ms	0.041 ms	0.086 ms	64 B	64 B
RSA-3072 (non-PQC)	400 ms	2.6 ms	0.027 ms	384 B	384 B

Table 2: Single-core median performance on Intel Xeon E-2124 3.3 GHz of some of NIST PQC signature algorithm candidates (and some current non-PQC alternatives) at NIST PQC security level 1 (Dilithium is at that submission's smallest suggested parameter set – at level 2). Source: r24000, supercop-20210604 at [103]. The measurements for SPHINCS are for a separate platform: Intel XeonE3-1220 3.1 GHz in Table 6 of [106]. LMS is a stateful hash-based signature scheme (see Section 4.4), these schemes have slow key generation, while signing and verification takes at most a few milliseconds on a comparable platform to those used by the other algorithms in the table.

From: <https://arxiv.org/pdf/2112.00399.pdf>

PQC not a “drop-in” replacement

- As you can see from the size/performance numbers, PQC algs don't satisfy the same size/CPU requirements met by classic public key algs
- Size/CPU impact varies from alg to alg and application to application:
 - BIG public keys or sigs – bad for DNSSEC
 - Bigger KEM outputs – not great but maybe ok if e.g. TLS ClientHello still fits in one packet
 - CPU performance for finalists seems ok, not true for all candidate algs though
- We still need more time to be more confident in these algorithms though

Kyber (1)

- Seems likely to see real use “soon”
- Based on module learning with errors (MLWE) problem
 - Can be stated a different ways, e.g. the closest vector problem (CVP): find the nearest lattice point (in n-dimensions) to a given point
- Variants: Kyber512, Kyber768, Kyber1024
- Kyber512
 - similar security level to AES128 (if a CRQC exists)
 - public key: 800 octets, encap: 768 octets
- Expected changes before standardisation will affect interop

Kyber (2)

- Note: I don't find this and the following slides satisfactory but they try to give a sense of how Kyber works...
- Arithmetic deals with vectors and matrices of polynomials over a ring
- R_q is $\mathbb{Z}_q[X]/(X^n + 1)$ for $n = 256$, $q = 3329$
- Secret vector \mathbf{s} is an element of R_q^2 and error vector \mathbf{e} another element of R_q^2
- Secret and (all) error vectors are “small” (think of ‘em as vectors of polynomials with small coefficients)
- We use a matrix \mathbf{A} from $R_q^{2 \times 2}$
- Public key is \mathbf{A} and $\mathbf{t} = \mathbf{As} + \mathbf{e}$

Kyber (3)

- Encrypt message m (< 256 bits) as a polynomial with one message bit per coefficient
- Random “small” vector \mathbf{r} in R_q^2 and error vectors $\mathbf{e}_1, \mathbf{e}_2$ also in \mathbf{R}_q^2
- $\mathbf{u} = \mathbf{A}\mathbf{r} + \mathbf{e}_1$
- $\mathbf{v} = \mathbf{t}\mathbf{r} + \mathbf{e}_2 + [\mathbf{q}/2]\mathbf{m}$
- Ciphertext is \mathbf{u}, \mathbf{v}
- In real alg, there’s a compression thing going on and a number of the values are represented differently

Kyber (4)

- To decrypt calculate $\mathbf{v} - \mathbf{s}\mathbf{u}$
$$= (\mathbf{A}\mathbf{s} + \mathbf{e})\mathbf{r} + \mathbf{e}_2 + [\mathbf{q}/2]\mathbf{m} - \mathbf{s}(\mathbf{A}\mathbf{r} + \mathbf{e}_1)$$
$$= \mathbf{A}\mathbf{s}\mathbf{r} + \mathbf{e}\mathbf{r} + \mathbf{e}_2 + [\mathbf{q}/2]\mathbf{m} - \mathbf{A}\mathbf{s}\mathbf{r} - \mathbf{s}\mathbf{e}_1$$
$$= [\mathbf{q}/2]\mathbf{m} + \mathbf{e}\mathbf{r} + \mathbf{e}_2 - \mathbf{s}\mathbf{e}_1$$
- Other than $[\mathbf{q}/2]\mathbf{m}$ all terms are “small” and since each \mathbf{m} coefficient is 0 or 1 result has coefficients that are either close to 0 in which case the relevant message bit was zero, or close to $[\mathbf{q}/2]$ in which case the relevant message bit was 1
- Why is this secure? Honestly, I’m not sure I understand it enough to say. Other more qualified people do think it secure though
- These slides describe the Kyber public key encryption alg, the KEM is a little different

Hash-Based Signatures

- These are **stateful** signature schemes which means that the private state (i.e. the value of the private key) changes with every signature operation
- Yes, they are quantum resistant, so worth exploring, esp. for things like software-signing where we won't need many signatures and sizes aren't a big deal
- **BUT they need a different cryptographic API** as private keys have a fixed, limited number of uses allowed, (maybe 2^{10} or 2^{20}) and if you go over that limit, you lose all security (signatures could be forged)
- Also, **if you EVER re-use a private key value you're hosed**, so you need to be very very careful e.g. about reboots, forking, and state maintenance
- Two algorithms standardised: XMSS (RFC 8391) and LMS (RFC 8554)
- You can't just replace Ed25519 with XMSS or LMS without changes to applications/systems

Embed PQ KEMs with classic DH (hybrid approach)

- Likely TLS, IPsec and other “classic” key exchanges will be extended to allow mixing in of PQ key exchanges **in addition** to the classic DH exchange
- Efficiency and scaling issues are tricky and not worked out yet really, and there could be IPR on some ways of mixing stuff in
- Bottom line: wait a while more to start using this, but getting closer, and if your data is long-term sensitive start experiments now
- Draft specification:
<https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/>

Quantum Key Distribution

- QKD: **Directly** exchange photons between endpoints e.g. (over a single-hop fibre or RF link); accumulate key bits based on received photon state (e.g. polarisation)
 - Claims made this is “unbreakable... because <physics>”
 - (At least) Saturation attacks exist against real implementations
 - <https://www.nature.com/articles/s41598-021-87574-4>
- Requires (classic) authentication of eventual key bits selected
 - <https://www.ietf.org/proceedings/99/slides/slides-99-saag-post-quantum-cryptography-01.pdf>
 - Slides 12-18
- By definition, QKD requires no intermediaries hence has scaling problems
- Still could be useful for specific links used for specific purposes
- But: be aware of marketing hype

Top Level Message

- Don't panic, most people don't need to care now
- Cryptographers are studying new but fairly well-defined algorithms that aim to provide good security even in the face of a cryptographically relevant quantum computer (CRQC)
- Engineers are starting to consider how to integrate those algorithms into protocols and applications
- Could well be that hybrid cryptographic solutions (using both well-understood “classic” algorithms and new, less well-understood, post-quantum algorithms) will be deployable in the next few years
 - Note “hybrid” isn't the best terminology here as we'll see next when we look at HPKE (RFC9180)