Post-Quantum Cryptography

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
- Engineers are starting to consider how to integrate those algorithms into protocols and applications
- Could well be that hybrid cryptographic solutions (combining currently well-understood “classic” algorithms with new, less well-understood post-quantum algorithms) will be deployable in the next few years

Resources

- We don’t 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

Overview

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

Quantum Computing

- Classical computers (today’s) based on binary bits
- Quantum computers based on (coherent) qubits that allow running algorithms that explore all different values at once
- Many, many well-resourced people are working to develop quantum computers that would improve on current classic computers
  - There are many other uses for quantum computers, they’re not only for breaking crypto
- Cryptographically relevant quantum computers 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 device will be built to solve a particular problem
- Current hardware only supports small numbers of physical qubits
- Error-correction/de-coherence problem means 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 quick 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
  - Less easy 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?

- If a cryptographically relevant quantum computer were to exist in 20xx then data that is protected 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 are likely be usable hybrid approaches to protecting such data (more later)

NIST Competition

- Started in 2017, results “soon”
  - 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 exchange
  - Signatures less urgent and we have stateful sigs already (later)
- Likely not one “winner” but more than one for KEM and Signature
- Currently at Round 3 “finalists” stage (but with “alternates” too;-)
  - I’ve not done my own analysis of the pros/cons of those (from the POV of how easy/difficult they are, not WRT cryptographic strength where I’m not qualified)
  - https://en.wikipedia.org/wiki/NIST_Post-Quantum_Cryptography_Standardization#Finalists

KEM algorithm	Generate key	Encaps.	Decaps.	Public key size	Encaps. size
NTRU (ntruhps2048509)	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].

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.

PQC not a “drop-in” replacement

- As you can see from the size/performance numbers, PQC algs don’t meet the same requirements met by classic public key algs
- Size/perf impact varies from 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
- There may be IPR issues – despite submitters of candidates declaring all known IPR, there are worries about 3rd parties maybe with patents

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 single signature operation
- Yes, they are quantum resistant, so worth exploring, esp for things like software-signing where we won’t need too many signatures and sizes aren’t a big deal
- BUT they need a different cryptographic API as those 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 and state maintenance
- So you can’t just replace Ed25519 with XMSS or LMS without changes to applications/systems
- Two algorithms standardised: XMSS (RFC 8391 and LMS (RFC 8554)
- Could be tricky when forking/spawning a process or VM

Embed PQ exchanges with classic DH

- 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
- This may involve betting on which algorithms you think may “win” the NIST competition (or just picking those you like)
- One VPN operator I chatted with a while back was interested in embedding 3 different PQ key exchanges into wireguard (an IPsec alternative) as a hedge against future quantum attacks
- Efficiency and scaling issues are tricky and not worked out yet really, and there’s IPR on some ways of mixing stuff in
- Bottom line: wait a while more to start on this! (but getting closer)
- Draft specification: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/

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
- Engineers are starting to consider how to integrate those algorithms into protocols and applications
- Could well be that hybrid cryptographic solutions (combining currently well-understood “classic” algorithms with new, less well-understood post-quantum algorithms) will be deployable in the next few years