The quantum clock is ticking... Get ready!

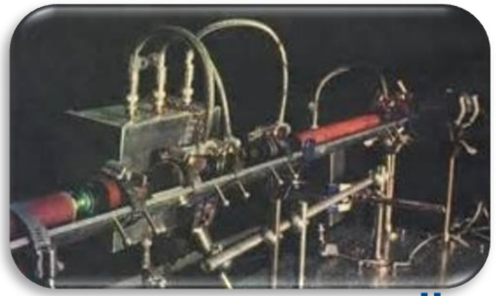
Christian Paquin

@chpaquin

Microsoft Research

nsec 2023

About me

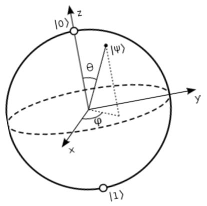


Université de Montréal

- Studied quantum cryptography 25+ years ago at University of Montreal
- Worked in the industry as a cryptographic engineer
- Now with the *MSR Security & Crypto* team, working on
  - Privacy-preserving identity
  - Post-Quantum Cryptography
  - Emerging cryptography
- Links
  - MSR page: https://www.microsoft.com/en-us/research/people/cpaquin/
  - Blog: https://christianpaquin.github.io/

The Quantum Revolution

- Quantum computers use the properties of *quantum mechanics* to implement algorithms not possible on classical computers



$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

- A lot of R&D around the globe
  - My colleagues are building the full stack: from the chip to the SDK https://www.microsoft.com/quantum/

QDK

Q#

Impact on Cryptography

- Shor solves the factoring (breaks RSA) and discreet log (breaks DSA, Diffie Hellman, and elliptic curve variants) problems in polynomial time
  - Grover improves attacks on symmetric cryptography (e.g., AES, SHA), but we have the solution: double the key/hash size

The Quantum Menace

- TL;DR: Breaks the asymmetric crypto in use today

https:// SSH

- Could be built within 10-15 years

- We need new *quantum-resistant* cryptography

NIST PQC standardization

- **Dec 2017**
  - Round 1
  - 69 submissions
- **Jan 2018**
  - Round 2
  - 26 algs
- **July 2020**
  - Round 3
  - 7 finalists
  - 8 alternates
- **July 2022**
  - 4 selected algs
  - Round 4 with 3 more algs
- YOU ARE HERE
- **June 2023**
  - Signature RFP
- **2024**
  - Draft standards for public comments

- Academia & industry has been working on the transition for years
- NIST started a PQC standardization project in 2017
- First PQC cryptography algorithms have been selected
- Draft standards expected in 2024
- https://csrc.nist.gov/Projects/post-quantum-cryptography/

Meet the new PQC algorithms

Key Encapsulation Mechanism (KEM) = encryption
Signature
Preferred by NIST

Crystals-Kyber

- Strong security and performance
- Three variants:
    - Kyber512 (L1)
    - Kyber768 (L3)
    - Kyber1024 (L5)

Crystals-Dilithium

- Security, high efficiency, simple implementation
- Three variants:
    - Dilithium2 (L1)
    - Dilithium3 (L3)
    - Dilithium5 (L5)

Falcon

- Small bandwidth, fast verification, but more complicated than Dilithium
- Two variants:
    - Falcon-512 (L1)
    - Falcon-1024 (L5)
- Standard after Dilithium

FALCON
Fast-Fourier Lattice-based Compact Signatures over NTRU

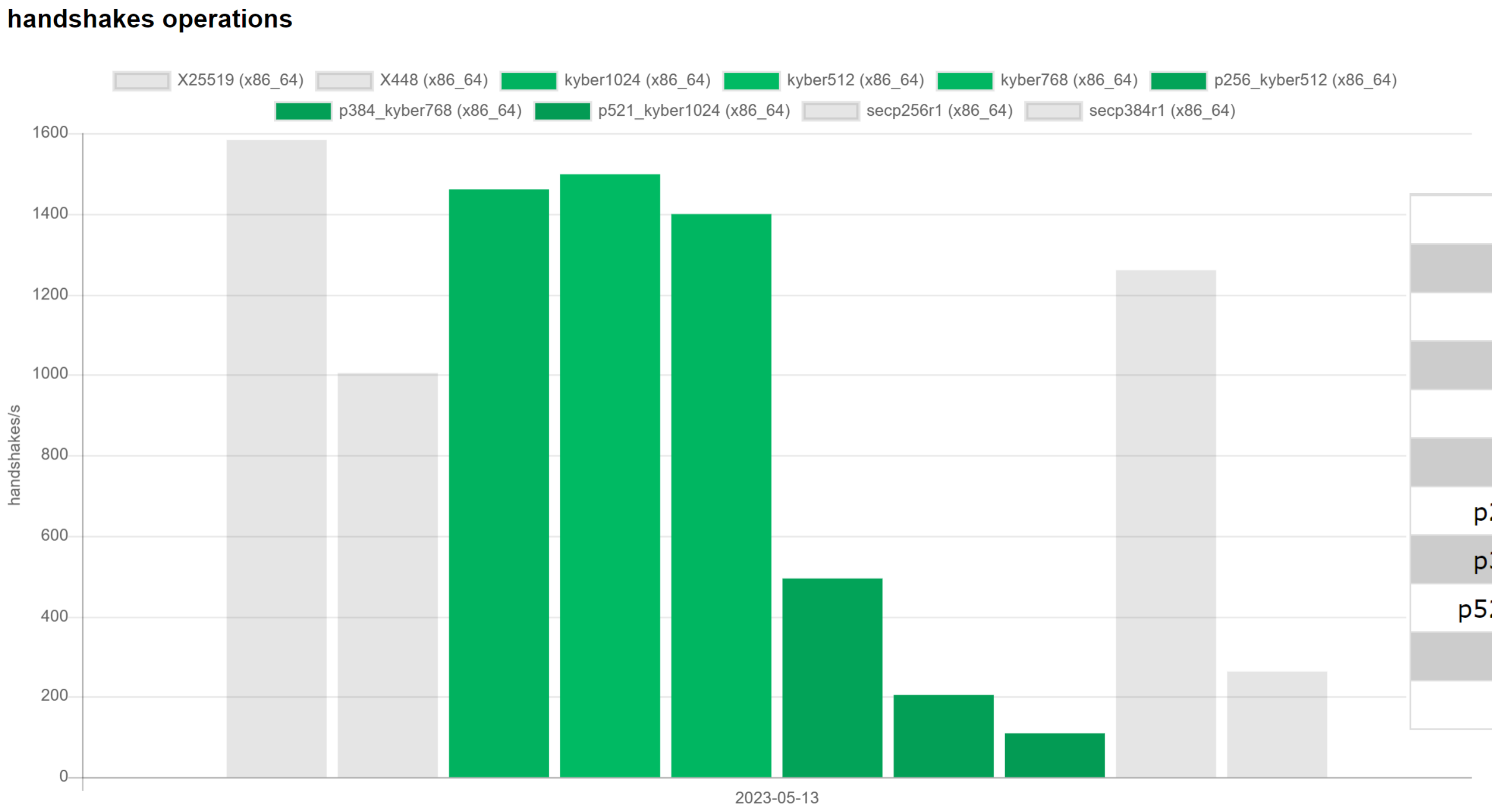
Sphincs+

- Solid security, hash-based
- Very large signatures
- Many parameter sets!
    - NIST to standardize L1, L3, and L5
    - Looking for feedback

SPHINCS+
Stateless hash-based signatures

Security Levels
L1 = AES128
L3 = AES192
L5 = AES256

Kyber vs. classical TLS performance



Algorithm	handshakes/s
X25519 (x86_64)	1583.05
X448 (x86_64)	1003.60
kyber1024 (x86_64)	1459.83
kyber512 (x86_64)	1496.67
kyber768 (x86_64)	1398.40
p256_kyber512 (x86_64)	494.21
p384_kyber768 (x86_64)	203.23
p521_kyber1024 (x86_64)	107.78
secp256r1 (x86_64)	1259.48
secp384r1 (x86_64)	262.75

https://openquantumsafe.org/benchmarking/visualization/handshakes.html

Size comparison with classical algorithms

Key Encapsulation Mechanisms (KEM)

Level	Algorithm	Secret Ket	Public Key	Ciphertext
L1	ECDHE P256	32	65	64
	Kyber512	1632	800	768
L3	ECDHE P384	48	97	96
	Kyber768	2400	1184	1088
L5	ECDHE P521	66	133	132
	Kyber1024	3168	1568	1568

Notes:
- All sizes in bytes
- Classical in red, PQC in black

Signatures

Level	Algorithm	Secret Key	Public Key	Signature
L1	ECDSA P256	32	65	64
	Dilithium2	2528	1312	2420
	Falcon512	1281	897	690
	Sphincs+ 128*	64	32	7856/17088
L3	ECDSA P384	48	97	96
	Dilithium3	4000	1952	3293
	Sphincs+ 192*	96	48	16224/35664
L5	ECDSA P521	66	133	132
	Dilithium5	4864	2592	4595
	Falcon1024	2305	1793	1330
	Sphincs+ 256*	128	64	29792/49856

3-step plan to get ready

1. Make your crypto inventory
2. Make sure you are agile
3. Start experimenting with PQC

Discovery of vulnerable crypto

- Check for vulnerable algorithms
  - In code
  - In protocols
  - On the wire
  - In software libraries
  - In the supply chain
- One useful tool for developers: CodeQL
  - CodeQL queries for PQC: https://github.com/raulgarciamsft/ql/tree/nccoe-pqv/cpp/ql/src/experimental/campaigns/nccoe-pqc-migration/QuantumVulnerableDiscovery

OPEN QUANTUM SAFE

- Development and prototyping of quantum-resistant cryptography
- *liboqs*: C library offering all NIST finalists and selected algorithms
- Bindings for [C++](), [C#](), [go](), [java](), [python](), [rust]()
- Protocol integration into TLS and CMS (OpenSSL [1.1.1]() and [3.0]()), SSH ([OpenSSH]() and [libssh]())
- [Application integration]() into curl, chromium, httpd, nginx, openvpn, quic, wireshark, and more
- Supports hybrid deployments (classical + PQC)
- [https://openquantumsafe.org]()

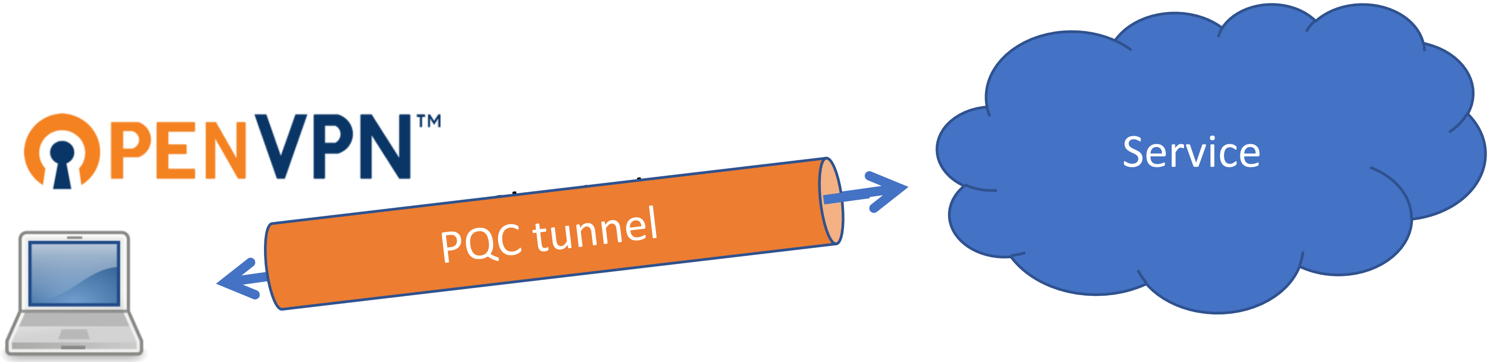
UNIVERSITY OF WATERLOO

Financial and in-kind support:

aws · NSERC CRSNG · cisco · Canadian Centre for Cyber Security | Centre canadien pour la cybersécurité · evolutionQ · Microsoft · IBM Research · Unitary Fund · NGI Assure · VERISIGN

PQ VPN tunnels

- OpenVPN integration
  - Uses OQS’s OpenSSL fork
  - Easy legacy app tunneling
  - https://www.microsoft.com/en-us/research/project/post-quantum-crypto-vpn/



- Project Natick PQC VPN experiment
  - Natick was an underwater datacenter module off the coast of Scotland
  - We ran a PQ VPN from Redmond
    - Used ECDHE-P256 + SIKEp434 hybrid
  - https://www.microsoft.com/en-us/research/project/post-quantum-crypto-tunnel-to-the-underwater-datacenter/

- Migration to Post-Quantum Cryptography project
- Organized by NIST’s National Cybersecurity Center of Excellence (NCCoE)
- Partnership with industry partners
- Goals:
  - Demonstrate vulnerable cryptography detection
  - Demonstrate PQC experimentation

- Amazon Web Services, Inc. (AWS)
- Cisco Systems, Inc.
- Cloudflare, Inc.
- Crypto4A Technologies, Inc.
- CryptoNext Security
- Dell Technologies
- DigiCert
- Entrust
- IBM
- Information Security Corporation
- InfoSec Global
- ISARA Corporation
- JPMorgan Chase Bank, N.A.
- Microsoft
- PQShield
- Samsung SDS Co., Ltd.
- SandboxAQ
- Thales DIS CPL USA, Inc.
- Thales Trusted Cyber Technologies
- VMware, Inc.
- wolfSSL

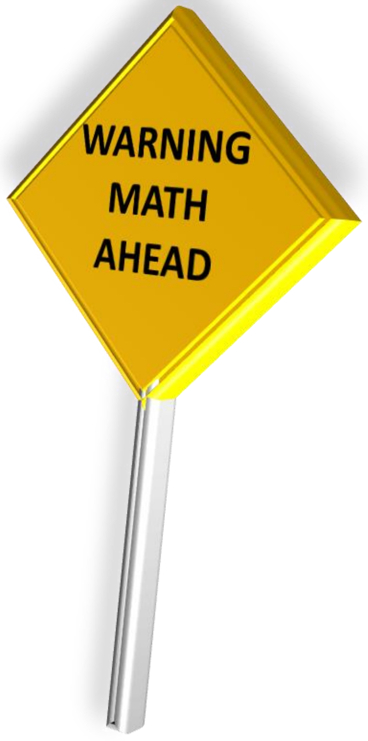
- https://www.nccoe.nist.gov/crypto-agility-considerations-migrating-post-quantum-cryptographic-algorithms

Quantum computers are coming…

Get ready for the PQC transition

1. Make your crypto inventory
2. Make sure you are agile
3. Start experimenting with PQC

Kyber toy example



Kyber toy example – setup

- Public parameters: modulo $q = 17$, polynomial $f = x^4 + 1$
- Secret key $\mathbf{s} = (-x^3 - x^2 + x, -x^3 - x)$
- Public key $(\mathbf{A}, \boldsymbol{t})$

$$\mathbf{A} = \begin{bmatrix} 6x^3 + 16x^2 + 16x + 11 & 9x^3 + 4x^2 + 6x + 3 \\ 5x^3 + 3x^2 + 10x + 1 & 6x^3 + x^2 + 9x + 15 \end{bmatrix}$$

$$\boldsymbol{e} = \left(x^2, x^2 - x\right)$$

$$\boldsymbol{t} = \mathbf{A}\mathbf{s} + \boldsymbol{e} = \left(16x^3 + 15x^2 + 7, 10x^3 + 12x^2 + 11x + 6\right)$$

https://cryptopedia.dev/posts/kyber

Kyber toy example – encryption

- Randomizer $\boldsymbol{r} = \left(-x^3 + x^2, x^3 + x^2 - 1\right)$
  Random error vector $\boldsymbol{e_1} = \left(x^2 + x, x^2\right)$
  Random error polynomial $e_2 = -x^3 - x^2$
- Message $= 1011, m_b = 1x^3 + 0x^2 + 1x^1 + 1x^0 = x^3 + x + 1$
- Scale message $m = \left\lceil \frac{q}{2} \right\rfloor m_b = 9\left(x^3 + x + 1\right) = 9x^3 + 9x + 9$
- Encrypt message
  $\boldsymbol{u} = \boldsymbol{A}^T\boldsymbol{r} + \boldsymbol{e_1} = \left(11x^3 + 11x^2 + 10x + 3, 4x^3 + 4x^2 + 13x + 11\right)$
  $v = \boldsymbol{t}^T\boldsymbol{r} + e_2 + m = 7x^3 + 6x^2 + 8x + 15$

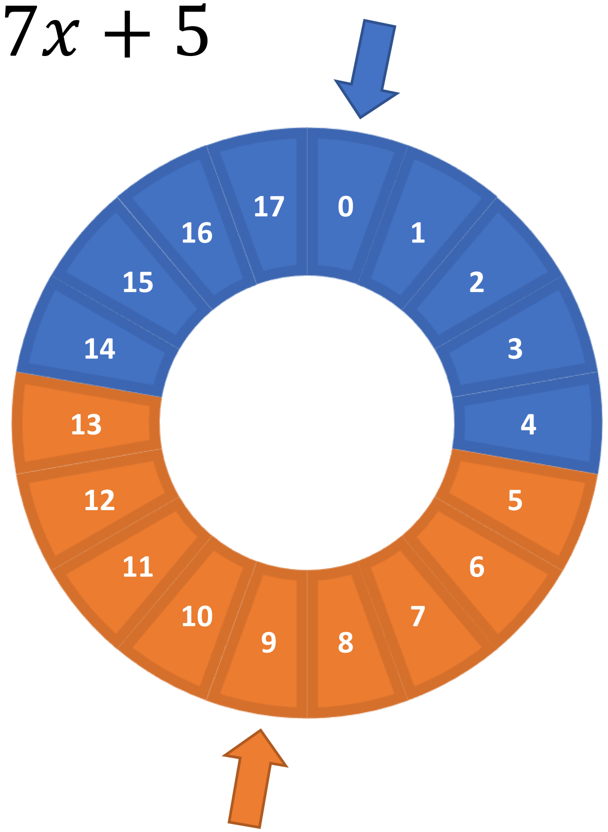
https://cryptopedia.dev/posts/kyber

Kyber toy example – decryption

- Noisy message $\mathrm{m_n} = v - \boldsymbol{s}^T\boldsymbol{u} = 7x^3 + 14x^2 + 7x + 5$
- Round each coefficient to 0 or $\left\lceil \frac{q}{2} \right\rfloor = 9$

  $m_n \rightarrow m = 9x^3 + 0x^2 + 9x + 9$
- Scale message back

  $$m_b = \frac{1}{9}m = 1x^3 + 0x^2 + 1x + 1 = 1011$$



https://cryptopedia.dev/posts/kyber

Full size Kyber

- Similar to the toy example, but with bigger parameters and compression

Name	n: **Max polynomial degree**	$\boldsymbol{k}$: **No of polynomials per vector**	**Modulus** q	η_1:max coefficient for small polynomials	$\boldsymbol{\eta_2}$:max coefficient for small polynomials	$\boldsymbol{d_u}$:**compression for** $\boldsymbol{u}$	$\boldsymbol{d_v}$:**compression for** $\boldsymbol{v}$	$\boldsymbol{\delta}$: **decryption error probability**
Kyber512	256	2	3329	3	2	10	4	$\frac{1}{2^{139}}$
Kyber768	256	3	3329	2	2	10	4	$\frac{1}{2^{164}}$
Kyber1024	256	4	3329	2	2	11	5	$\frac{1}{2^{174}}$

https://cryptopedia.dev/posts/kyber