Post-Quantum Cryptography

Peter Schwabe

Max Planck Institute for Security and Privacy

September 4, 2025

Disclaimer: no quantum physics

Klaus: *"Wir würden uns sehr freuen, Dich dort als Redner (Thema: Quantum Cryptography) begrüßen zu dürfen."*

Peter: *"Vielen herzlichen Dank für die Einladung! Ich würde schon gerne dort einen Vortrag halten, aber ich mache ja keine Quantenkryptographie, sondern Post-Quanten Kryptographie."*

Klaus: *"Ja, stimmt "Post-Quantum-Kryptographie", umso besser. ;-) Freue mich über Deine Zusage, Titel ändern wir."*

[A small demo]

ECDH and X25519

Let G be a finite cyclic group with generator g.

Alice		Bob
$A \leftarrow g^a$		$B \leftarrow g^b$
	A →	
	← B	
$K \leftarrow B^a = (g^b)^a = g^{ab}$		$K \leftarrow A^b = (g^a)^b = g^{ab}$

ECDH and X25519

- Diffie, Hellman, 1976: Use $G = GF(q)^*$
- Miller, Koblitz (independently), 1985/86: Use group of points on an elliptic curve
- Bernstein, 2006: Use specific elliptic curve over $GF(2^{255} - 19)$

(EC)DH is everywhere

Nitrokey
secure your digital life

WireGuard
FAST, MODERN, SECURE VPN TUNNEL

Tor

zoom

The Discrete Logarithm Problem

Definition

Given $P, Q \in G$ such that $Q \in \langle P \rangle$, find an integer k such that $P^k = Q$.

The Discrete Logarithm Problem

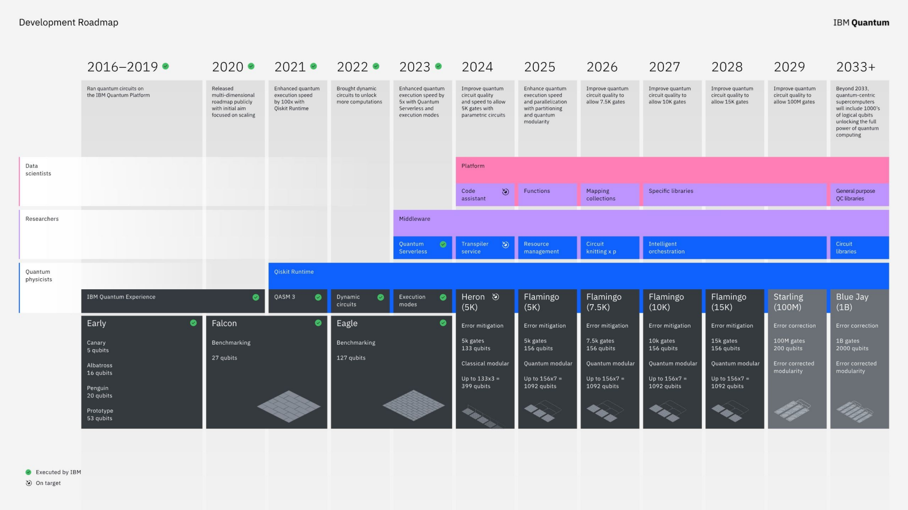
- DH needs group where DLP is hard
- (EC)DLP-based crypto also for signatures (DSA, ECDSA, EdDSA. . .)
- Prominent alternative: RSA (based on factoring)

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.



See https://www.ibm.com/quantum/blog/ibm-quantum-roadmap-2025

Post-quantum crypto (PQC)

Definition

Post-quantum crypto is (asymmetric) crypto that resists attacks using classical *and quantum* computers.

Post-quantum crypto (PQC)

Definition

Post-quantum crypto is (asymmetric) crypto that resists attacks using classical *and quantum* computers.

5 main directions

- Lattice-based crypto (PKE and Sigs)
- Code-based crypto (mainly PKE)
- Multivariate-based crypto (mainly Sigs)
- Hash-based signatures (only Sigs)
- Isogeny-based crypto (it's complicated. . .)

Should you care now?

"Harvest now, decrypt later"

https://en.wikipedia.org/wiki/Utah_Data_Center#/media/File:EFF_photograph_of_NSA's_Utah_Data_Center.jpg

Should you care now?

“Harvest now, decrypt later”

https://en.wikipedia.org/wiki/Utah_Data_Center#/media/File:EFF_photograph_of_NSA's_Utah_Data_Center.jpg

Mosca’s theorem

$$X + Y > Z$$

- X: For how long do you need encrypted data to be secure?
- Y: How long does it take you to migrate to PQC
- Z: Time it will take to build a cryptographically relevant quantum computer

If $X + Y > Z$, you should worry.

NIST PQC – how it started

Count of Problem Category	Column Labels		
Row Labels	Key Exchange	Signature	Grand Total
?	1		1
Braids	1	1	2
Chebychev	1		1
Codes	19	5	24
Finite Automata	1	1	2
Hash		4	4
Hypercomplex Numbers	1		1
Isogeny	1		1
Lattice	24	4	28
Mult. Var	6	7	13
Rand. walk	1		1
RSA	1	1	2
Grand Total	**57**	**23**	**80**

4 31 27

Overview tweeted by Jacob Alperin-Sheriff on Dec 4, 2017.

NIST PQC – how it went

NIST PQC

Nov. 2017
69 proposals
→ Round 1 →
Feb. 2019
26 proposals
→ Round 2 →
Jul. 2020
7+8 proposals
→ Round 3 →
Jul. 2022
4 "winners"

NIST PQC – how it went

NIST PQC

Nov. 2017 69 proposals	Round 1 →	Feb. 2019 26 proposals	Round 2 →	Jul. 2020 7+8 proposals	Round 3 →	Jul. 2022 **4 "winners"**

"The public-key encryption and key-establishment algorithm that will be standardized is ***CRYSTALS-KYBER****. The digital signatures that will be standardized are CRYSTALS-Dilithium, FALCON, and SPHINCS*$^+$*. While there are multiple signature algorithms selected, NIST recommends* ***CRYSTALS-Dilithium*** *as the primary algorithm to be implemented"*

—NIST IR 8413-upd1

[Back to our demo]

Key Encapsulation Mechanisms (KEMs)

Initiator		Responder
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM.Gen}$		
	$\xrightarrow{\mathsf{pk}}$	
		$(\mathsf{ct}, K) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk})$
	$\xleftarrow{\mathsf{ct}}$	
$K \leftarrow \mathsf{KEM.Dec}(\mathsf{ct}, \mathsf{sk})$		

Learning with errors (LWE)

- Given uniform $\mathbf{A} \in \mathbb{Z}_q^{k \times \ell}$
- Given "noise distribution" χ
- Given samples $\mathbf{As} + \mathbf{e}$, with $\mathbf{e} \leftarrow \chi$

Learning with errors (LWE)

- Given uniform $\mathbf{A} \in \mathbb{Z}_q^{k \times \ell}$
- Given "noise distribution" χ
- Given samples $\mathbf{As} + \mathbf{e}$, with $\mathbf{e} \leftarrow \chi$
- Search version: find $\mathbf{s}$
- Decision version: distinguish from uniform random

Ring Learning with errors (RLWE)

- Given uniform $\mathbf{a} \in \mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$
- Given "noise distribution" χ
- Given samples $\mathbf{as} + \mathbf{e}$, with $\mathbf{e} \leftarrow \chi$

Ring Learning with errors (RLWE)

- Given uniform $\mathbf{a} \in \mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$
- Given "noise distribution" χ
- Given samples $\mathbf{as} + \mathbf{e}$, with $\mathbf{e} \leftarrow \chi$
- Search version: find $\mathbf{s}$
- Decision version: distinguish from uniform random

Alice (server)		Bob (client)
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{a}\mathbf{s} + \mathbf{e}$	$\xrightarrow{\mathbf{b}}$	$\mathbf{u} \leftarrow \mathbf{a}\mathbf{s}' + \mathbf{e}'$
	$\xleftarrow{\mathbf{u}}$	

Alice has $\mathbf{v} = \mathbf{u}\mathbf{s} = \mathbf{a}\mathbf{s}\mathbf{s}' + \mathbf{e}'\mathbf{s}$

Bob has $\mathbf{v}' = \mathbf{b}\mathbf{s}' = \mathbf{a}\mathbf{s}\mathbf{s}' + \mathbf{e}\mathbf{s}'$

- Secret and noise polynomials $\mathbf{s}, \mathbf{s}', \mathbf{e}, \mathbf{e}'$ are small
- $\mathbf{v}$ and $\mathbf{v}'$ are *approximately* the same

How to build a KEM, part 2

Alice		Bob
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{a}\mathbf{s} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}\quad)}$	
		$\mathbf{u} \leftarrow \mathbf{a}\mathbf{s}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{b}\mathbf{s}'$
$\mathbf{v}' \leftarrow \mathbf{u}\mathbf{s}$	$\xleftarrow{(\mathbf{u}\quad)}$	

How to build a KEM, part 2

Alice		Bob
$seed \xleftarrow{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$		
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$
		$\mathbf{u} \leftarrow \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{bs}'$
$\mathbf{v}' \leftarrow \mathbf{us}$	$\xleftarrow{(\mathbf{u}\quad)}$	

How to build a KEM, part 2

Alice		Bob
$seed \xleftarrow{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$		
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$
		$\mathbf{u} \leftarrow \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{bs}'$
		$k \xleftarrow{\$} \{0,1\}^n$
		$\mathbf{k} \leftarrow \text{Encode}(k)$
$\mathbf{v}' \leftarrow \mathbf{us}$	$\xleftarrow{(\mathbf{u}, \mathbf{c})}$	$\mathbf{c} \leftarrow \mathbf{v} + \mathbf{k}$

How to build a KEM, part 2

Alice		Bob
$seed \xleftarrow{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$		
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$
		$\mathbf{u} \leftarrow \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{bs}' + \mathbf{e}''$
		$k \xleftarrow{\$} \{0,1\}^n$
		$\mathbf{k} \leftarrow \text{Encode}(k)$
$\mathbf{v}' \leftarrow \mathbf{us}$	$\xleftarrow{(\mathbf{u}, \mathbf{c})}$	$\mathbf{c} \leftarrow \mathbf{v} + \mathbf{k}$

How to build a KEM, part 2

Alice		Bob
$seed \xleftarrow{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$		
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{a}\mathbf{s} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$
		$\mathbf{u} \leftarrow \mathbf{a}\mathbf{s}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{b}\mathbf{s}' + \mathbf{e}''$
		$k \xleftarrow{\$} \{0,1\}^n$
		$\mathbf{k} \leftarrow \text{Encode}(k)$
$\mathbf{v}' \leftarrow \mathbf{u}\mathbf{s}$	$\xleftarrow{(\mathbf{u}, \mathbf{c})}$	$\mathbf{c} \leftarrow \mathbf{v} + \mathbf{k}$
$\mathbf{k}' \leftarrow \mathbf{c} - \mathbf{v}'$		

How to build a KEM, part 2

Alice		Bob
$seed \xleftarrow{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$		
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$
		$\mathbf{u} \leftarrow \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{bs}' + \mathbf{e}''$
		$k \xleftarrow{\$} \{0,1\}^n$
		$\mathbf{k} \leftarrow \text{Encode}(k)$
$\mathbf{v}' \leftarrow \mathbf{us}$	$\xleftarrow{(\mathbf{u}, \mathbf{c})}$	$\mathbf{c} \leftarrow \mathbf{v} + \mathbf{k}$
$\mathbf{k}' \leftarrow \mathbf{c} - \mathbf{v}'$		$\mu \leftarrow \text{Extract}(\mathbf{k})$
$\mu \leftarrow \text{Extract}(\mathbf{k}')$		

How to build a KEM, part 2

Alice		Bob
$seed \xleftarrow{\$} \{0,1\}^{256}$		
$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$		
$\mathbf{s}, \mathbf{e} \xleftarrow{\$} \chi$		$\mathbf{s}', \mathbf{e}', \mathbf{e}'' \xleftarrow{\$} \chi$
$\mathbf{b} \leftarrow \mathbf{as} + \mathbf{e}$	$\xrightarrow{(\mathbf{b}, seed)}$	$\mathbf{a} \leftarrow \text{Parse}(\text{XOF}(seed))$
		$\mathbf{u} \leftarrow \mathbf{as}' + \mathbf{e}'$
		$\mathbf{v} \leftarrow \mathbf{bs}' + \mathbf{e}''$
		$k \xleftarrow{\$} \{0,1\}^n$
		$\mathbf{k} \leftarrow \text{Encode}(k)$
$\mathbf{v}' \leftarrow \mathbf{us}$	$\xleftarrow{(\mathbf{u}, \mathbf{c})}$	$\mathbf{c} \leftarrow \mathbf{v} + \mathbf{k}$
$\mathbf{k}' \leftarrow \mathbf{c} - \mathbf{v}'$		$\mu \leftarrow \text{Extract}(\mathbf{k})$
$\mu \leftarrow \text{Extract}(\mathbf{k}')$		

Encryption scheme by Lyubashevsky, Peikert, Regev. Eurocrypt 2010.

Encode and Extract

- Encoding in LPR encryption: map n bits to n coefficients:
  - A zero bit maps to 0
  - A one bit maps to $q/2$
- Idea: Noise affects low bits of coefficients, put data into high bits

Encode and Extract

- Encoding in LPR encryption: map n bits to n coefficients:
  - A zero bit maps to 0
  - A one bit maps to $q/2$
- Idea: Noise affects low bits of coefficients, put data into high bits
- Decode: map coefficient into $[-q/2, q/2]$
  - Closer to 0 (i.e., in $[-q/4, q/4]$): set bit to zero
  - Closer to $\pm q/2$: set bit to one

KYBER

Joint work with Avanzi, Bos, Ducas, Kiltz, Lepoint, Lyubashevsky, Schanck, Seiler, Stehlé, Ding

- LPR encryption using Module-LWE (variant of RLWE)
- Many tweaks, parameter optimizations, etc.

KYBER

Joint work with Avanzi, Bos, Ducas, Kiltz, Lepoint, Lyubashevsky, Schanck, Seiler, Stehlé, Ding

- LPR encryption using Module-LWE (variant of RLWE)
- Many tweaks, parameter optimizations, etc.
- Transform for active security
  - Make it safe to re-use keys for multiple executions
  - More flexible building block for protocols

KYBER

Joint work with Avanzi, Bos, Ducas, Kiltz, Lepoint, Lyubashevsky, Schanck, Seiler, Stehlé, Ding

- LPR encryption using Module-LWE (variant of RLWE)
- Many tweaks, parameter optimizations, etc.
- Transform for active security
  - Make it safe to re-use keys for multiple executions
  - More flexible building block for protocols
- In 2024 standardized as FIPS-203 (ML-KEM)
- Already in use by TLS, Signal, iMessage, AWS, OpenSSH. . .
- Secures several 100 billion connections per day at Cloudflare alone

MPI-SP: Construction status