Quantum Computing and Post-quantum Cryptography PART 2

André Schrottenloher

Inria Rennes
Team CAPSULE

Summary of Part 1

- Quantum computing (QC) is an enhanced **model of computation**
- ...in which **(only)** some problems admit significant **speedups**

Discrete Logarithms and factoring are solved in **polynomial time** with a large-scale QC.

$\Longrightarrow$ cannot be used as crypto "hard problems" anymore

- Unfortunately, this is (almost) **all** the currently deployed public-key crypto

- Large-scale QCs **do not yet exist**
- The commercial impact of QC might be overhyped, but the crypto threat is real

Outline

Preliminaries

Some history

- In 1976, Diffie and Hellman invent the principle of **public-key cryptography** and the **DH key-exchange mechanism**
- In 1978, Rivest, Shamir and Adleman invent the **RSA public-key cryptosystem**

  (British military cryptographers already knew about that, but all of their work remained classified until 1997)

But there's more to the story . . .

Diffie, Hellman, "New directions in cryptography", IEEE transactions on information theory, 1976

Rivest, Shamir, Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Commun. ACM 21.2, 1978

The McEliece cryptosystem (1978)

A Public-Key Cryptosystem Based On Algebraic Coding Theory

R. J. McEliece
Communications Systems Research Section

Using the fact that a fast decoding algorithm exists for a general Goppa code, while no such exists for a general linear code, we construct a public-key cryptosystem which appears quite secure while at the same time allowing extremely rapid data rates. This kind of cryptosystem is ideal for use in multi-user communication networks, such as those envisioned by NASA for the distribution of space-acquired data

- The scheme is based on error-correcting codes
- The main drawback is the **public key size**: millions of bits vs. thousands for RSA
- But it **resists quantum attacks!**

Post-quantum cryptography predates the theory of quantum computing!

McEliece, "A public-key system based on algebraic coding theory", DSN Progress Report 44, Jet Propulsion Lab, 1978

What we need

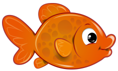
To secure the internet, we need:

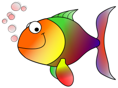
- Public-key encryption $\Longrightarrow$ **more urgent**
- Digital signatures $\Longrightarrow$ **less urgent**
- Secret-key (authenticated) encryption

  $\Longrightarrow$ we shouldn't need new constructions for this one

Public-key encryption (PKE)

$$\begin{cases} \text{KeyGen}: & 1^\lambda & \mapsto & \text{sk}, \text{pk} \\ \text{Encrypt}: & \text{m}, \text{pk} & \mapsto & \text{c} \\ \text{Decrypt}: & \text{c}, \text{sk} & \mapsto & \text{m} \end{cases}$$





$\text{sk}, \text{pk} = \text{KeyGen}(1^\lambda)$

$\xrightarrow{\text{pk}}$

$\text{c} = \text{Encrypt}(\text{m}, \text{pk})$

$\xleftarrow{\text{c}}$

$\text{m} = \text{Decrypt}(\text{c}, \text{sk})$

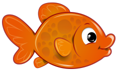
Typical attacks:

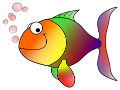
- key security: recover sk from pk
- message security: recover m from c

Key encapsulation mechanism (KEM)

$$\begin{cases} \text{KeyGen}: & 1^\lambda & \mapsto & \text{sk}, \text{pk} \\ \text{Encaps}: & \text{pk} & \mapsto & \text{c}, \text{k} \\ \text{Decaps}: & \text{sk}, \text{c} & \mapsto & \text{k} \end{cases}$$

$\text{k} \in \mathcal{K}$ is to be used as a symmetric secret key.





$\text{sk}, \text{pk} = \text{KeyGen}(1^\lambda)$

$\xrightarrow{\text{pk}}$

$\text{c}, \text{k} = \text{Encaps}(\text{pk})$

$\xleftarrow{\text{c}}$

$\text{k} = \text{Decaps}(\text{c}, \text{sk})$

Typical attacks:

- key security
- session key security: recover k from c

Hardness assumptions

- Breaking the scheme $\implies$ (via proof) solving some hard problem
- We study classical and quantum algorithms for the problem
- Knowing the best algorithms, we parameterize the scheme to make attacks infeasible

But sometimes:

- there is no proof
- the reduction does not land on the problem we want, but "close"
- the reduction exists, but is not tight
- the underlying problem happens to be not so hard

$\implies$ **cryptanalysis** has the final word*

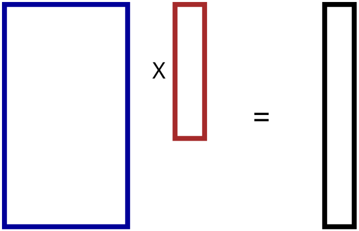
* Disclaimer: this statement comes from a cryptanalyst.

Lattice-based Cryptosystems (without the lattices)

Lattice-based crypto: summary

- At the moment, three lattice-based schemes are on their way to standardization by NIST: **Kyber** (ML-KEM), **Dilithium** (ML-DSA), **Falcon** (FN-DSA)
- Lattice-based schemes are solid and reach small parameter sizes (PK, SK, ciphertext)
- Compared to RSA, still **doubling** or **quadrupling** the PK size for equivalent security
- Also, implementing these schemes is harder than RSA, **but** the runtime is typically faster

A bad cryptosystem (do not use it!)



- Choose a public matrix $A \in \mathbb{Z}_q^{\ell \times n}$ at random
- Choose $s \in \mathbb{Z}_q^n$ at random: our private key
- Let (A, As) be our public key

Still a bad cryptosystem (do not use it!)

KeyGen:

- Private key: random $s \in \mathbb{Z}_q^n$
- Public key: random matrix A, $b := As$

Encrypt $m \in \{0,1\}$:

- Pick a random vector $r \in \{0,1\}^\ell$
- Return $c_1, c_2 := rA, (m + r \cdot b)$

Decrypt $(c_1, c_2) \in \mathbb{Z}_q^{n+1}$:

- Return $m = c_2 - c_1 \cdot s$

$$\begin{aligned} c_2 - c_1 \cdot s &= (m + r \cdot b) - (rA)s \\ &= m + (r \cdot b) - r(\underbrace{(As)}_{=b}) = m \end{aligned}$$

Why is this broken?

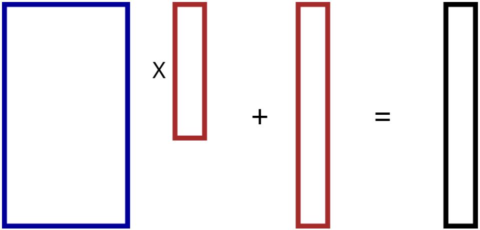
Let's do a Chosen-plaintext attack and always encrypt 0. We observe **samples:**

$$\mathsf{r}\mathsf{A}, (\mathsf{r}\mathsf{A}) \cdot \mathsf{s} \tag{1}$$

for unknown r and s.
After enough samples we have $\boldsymbol{R}, \boldsymbol{R}\mathsf{s}$: invert $\boldsymbol{R}$ to find s.

Linear algebra is not enough for crypto. We need another ingredient.

Learning with errors (LWE)



- Choose a public matrix $A \in \mathbb{Z}_q^{\ell \times n}$ at random
- Choose $s \in \mathbb{Z}_q^n$ at random **and** a "small" error $e \in \mathbb{Z}_q^\ell$
- Public key: $A, b := (As + e)$

Search: find s. **Decision:** distinguish the output from uniform.

We have good reasons to believe that LWE is hard*.

* *Quantum* reduction from average-case LWE to worst-case lattice problems.

Regev, "On lattices, learning with errors, random linear codes, and cryptography", STOC 2005

LWE: encryption scheme

Define:

- **Compress**: decodes an integer mod q into 0 if it's closer to 0 or 1 if it's closer to $q/2$
- **Decompress**: encodes 0 to 0 and 1 to $q/2$

KeyGen:

- Private key: random $s \in \mathbb{Z}_q^n$
- Public key: random matrix A, $b := As + e$ with e "small"

LWE: encryption scheme

Encrypt $m \in \{0,1\}$:

- Pick a random vector $r \in \{0,1\}^{\ell}$
- Return $c_1, c_2 := rA, (\text{Decompress}(m) + r \cdot b)$

Decrypt $(c_1, c_2) \in \mathbb{Z}_q^{n+1}$:

- $m = \text{Compress}(c_2 - c_1 \cdot s)$

Why this works:

$$\begin{aligned} c_2 - c_1 \cdot s &= (\text{Decompress}(m) + r \cdot b) - (rA) \cdot s \\ &= \text{Decompress}(m) + r(As + e) - rAs \\ &= \text{Decompress}(m) + \underbrace{r \cdot e}_{\text{Small}} \end{aligned}$$

The decryption may fail with some insignificant probability (e.g., 2^{-160} in Kyber according to the designers)

Limits of “basic” LWE

- The ciphertext is big: $\mathcal{O}(n)$ bits for only one bit encrypted
- The public key is big (around $\mathcal{O}(n^2)$ bits for n-bit security)
- The computation time is big (around $\mathcal{O}(n^2)$ for a matrix-vector product)

More algebra fixes it: we replace **matrices** over $\mathbb{Z}_q$ by **polynomials**.

Polynomial rings

Fix an integer n, and let: $\mathcal{R} = \mathbb{Z}[X]/(X^n + 1)$, i.e., polynomials where "$X^n = -1$".

$$\boldsymbol{a} = (a_0, \ldots, a_{n-1}) \leftrightarrow \boldsymbol{a}(X) = a_0 + a_1 X + \ldots + a_{n-1} X^{n-1}$$

$\Longrightarrow$ after all operations (addition, product ...) simply divide the polynomials by $X^n + 1$.

- We can also define such polynomials mod q: $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^n + 1)$

Ring-LWE

- Let's agree on a **distribution of polynomials** $D_{\mathcal{R}}$ with **small coefficients**

- Public $a \in \mathcal{R}_q$ chosen u.a.r.
- Secret $s, e \in \mathcal{R}$ chosen with $D_{\mathcal{R}}$
- Given $b = a \cdot s + e \pmod{q}$, find s (search) or distinguish from random (decision)

With this:

- We have n coefficients (vector a, b) instead of n^2 for a problem of size n!
- We will be able to encrypt n bits instead of 1
- We accelerate the encryption / decryption to $\mathcal{O}(n \log n)$ operations

Ring-LWE as in Kyber

KeyGen:

- Private key: $s \in \mathcal{R}_q, e \in \mathcal{R}_q$ sampled using $D_\mathcal{R}$ (i.e., small)
- Public key: random $a \in \mathcal{R}_q$, and $b := a \cdot s + e$

Encrypt $m \in \{0,1\}^n$ (as an element of $\mathcal{R}_q$)

- $r \xleftarrow{D_\mathcal{R}} \mathcal{R}_q$, $e_1 \xleftarrow{D_\mathcal{R}} \mathcal{R}_q$, $e_2 \xleftarrow{D_\mathcal{R}} \mathcal{R}_q$
- $c_1 := a \cdot r + e_1$
- $c_2 := b \cdot r + e_2 + \text{Decompress}(m)$
- Return (c_1, c_2)

Decrypt (c_1, c_2)

- $m = \text{Compress}(c_2 - s \cdot c_1)$

$$\begin{aligned} c_2 - s \cdot c_1 &= b \cdot r + e_2 + \text{Decompress}(m) - s \cdot (a \cdot r + e_1) \\ &= s \cdot a \cdot r + e \cdot r + e_2 + \text{Decompress}(m) - s \cdot a \cdot r - s \cdot e_1 \\ &= \underbrace{e \cdot r + e_2 - s \cdot e_1}_{\text{Small}} + \text{Decompress}(m) \ . \end{aligned}$$

Other Families

Summary

The post-quantum schemes are classified depending on (the objects underlying) their **hardness assumptions**.

Lattices

- SVP/SIS in random or structured lattices
- Mature, upcoming standards

Error-correcting codes

- Decoding random-looking codes
- Larger public key sizes
- Another KEM expected for standardization

Summary

Multivariate systems

- Solving random-looking mulivariate equation systems
- Give quite good signatures (you can expect one standard in the coming years)

Others

- Elliptic curve isogenies: small parameters & heavy computations, not mature
- Hash-based signatures: already standardized (SPHINCS+)
- ...

Code-based crypto

An **error-correcting code** is a way to encode information with redundancy, ex.:

Charlie, **O**scar, **D**elta, **E**cho

Formally we encode words of dimension k over $\mathbb{F}_2$ into codewords of dimension $n > k$ using a vector space defined by an $k \times n$ matrix.

Decoding problems:

- Given a noisy codeword $c = mG + e$ ("small" $e = t$ bit flips), find the original word m
- Given an error syndrome $s = He$, find the weight-t vector e

Decoding random codes is a hard problem

The McEliece scheme

Use a family of codes $\mathcal{F}$ that admit an efficient decoding algorithm, and a procedure to generate such codes:

$$\begin{cases} \mathcal{S} \to \mathcal{F} \\ s \mapsto \mathcal{C}(s) \end{cases}$$

KeyGen:

- Alice picks a secret key $\mathsf{s} \in \mathcal{S}$
- Reveal a parity-check matrix H of the code $\mathcal{C}(\mathsf{s})$

Encrypt: e

- Send $\mathsf{c} = \mathsf{He}$

Decrypt:

- Recover e using a fast decoding algorithm

Bonus: Secret-key Cryptography

Secret-key cryptography

Secret-key cryptography seems mostly* secure against QCs.

More precisely:

- in theory, we **could** have Shor-like speedups on classically secure ciphers / hash functions
- but these constructions are only theoretical

* Up to the cryptanalysis we tried so far

Yamakawa, Zhandry, "Verifiable Quantum Advantage without Structure", FOCS 2022

Example

The AES-256 block cipher: encrypts 128-bit "blocks" using a 256-bit secret key.

Security of an "ideal" block cipher:

- **classical:** try all the keys (number of keys = 2^{256})
- **quantum:** use Grover's algorithm

$\Longrightarrow$ generic reduction in security, but a manageable one: $2^{256/2} = 2^{128}$ remains infeasible

Is that all we can do? Nobody knows. We must try to cryptanalyze.*

* That's what I do.

Grover, "A fast quantum mechanical algorithm for database search", STOC 1996

At the moment

Block cipher key-recovery:

- generic: $T \to T^{1/2}$
- attacks have speedups between T^1 and $T^{1/2}$
- on **specific** (but realistic) designs: between $T^{1/2}$ and $T^{2/5}$

Hash function collision:

- generic: $T \to T^{2/3}$
- attacks have speedups between T^1 and $T^{1/2}$

Hash function preimage:

- generic: $T \to T^{1/2}$
- attacks have speedups between T^1 and $T^{1/2}$

Only in the "store now, decrypt later" attacker model. (There are weirder models.)

The NIST Process: a Brief Summary

The NIST process

Follows a long tradition of competitions: AES, SHA-3, CAESAR...
It's not only an American thing: NIST standards are *de facto* world standards.

- Bring together all researchers in the field
- Many teams propose their designs
- Let the best win!

https://csrc.nist.gov/projects/post-quantum-cryptography

The NIST process (ctd.)

- No one is safe from cryptanalysis
- Many (most?) of the candidates will be terribly broken
- Including some of your favorite

The NIST process (ctd.)

82 submissions for **KEMs** and **digital signatures**: $\simeq 64$ in the first round.

KEMs	Lattice	Codes	Multivariate	Other
Round 1 (2017)	21	17	2	5
Round 2 (2019)	9	7	0	0
Round 3 (2020) (finalists)	3	1	0	0
Round 3 (2020) (alternate)	2	2	0	1[a]
First standards (2022)	1[b]	0	0	0
Round 4 (2022)	0	3	0	0

a SIKE was broken

b CRYSTALS-Kyber based on Module-LWE

The NIST process (ctd.)

Signatures	Lattices	Codes	Multivariate	Other
Round 1	5	3	9	4
Round 2	3	0	4	2
Round 3 (2020) (finalists)	2	0	1[a]	0
Round 3 (2020) (alternate)	0	0	1[b]	2[c]
First standards (2022)	2[d]	0	0	1

- **a** Rainbow was broken
- **b** GeMSS was broken
- **c** SPHINCS+ and Picnic (only SPHINCS+ survived)
- **d** CRYSTALS-Dilithium and Falcon

There is no round 4 because no one else survived ☹

Thoughts on the NIST process

We learned a lot.*

- Lots of research is still ongoing on the **construction** side: many constructions are **not mature**
- NIST has made rather **conservative** choices, which were **justified**
- Even for the selected standards, lots of (applied) research remains on **secure implementations** and **integration**

* Euphemism: many hopes were broken, many tears were shed.

Bonus track: the NIST PQC-DS process

In Sept. 2023 NIST launched an additional call for signatures.

	Codes	Isogenies	Lattices	MPCitH	Multivar	Symm.	Other
Round 1	6	1	7	7	10	4	5
Now	4	1	5	7	7	4	2

Conclusion

On post-quantum crypto

- Even mature solutions need work on implementations and hardware
- Many alternatives, but many of them are not mature (i.e., should not be used in production)

Don't roll your own crypto, post-quantum edition:

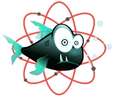
- many designs were broken in the NIST process
- even those of teams with high expertise

$$\text{Security} = \int_0^t \text{cryptanalysis}$$

Recommendations

- National agencies like ANSSI (France) or BSI (Germany) have made recommendations for post-quantum crypto & transition timelines
- Some of them may recommend schemes which are not NIST standards, but were put to the test (e.g., FrodoKEM)
- Most of them recommend **hybrid encryption** (pre + post-quantum) in the near future

Thank you!



BSI, "Quantum-safe cryptography - fundamentals, current developments and recommendations",

"ANSSI Views on the Post-Quantum Cryptography transition" March 25, 2022

TNO, CWI, AIVD, "The PQC migration handbook", 2023