Post-quantum cryptography

~antoniofrighetto

Introduction

Introduction

- Quantum computers take advantage of quantum mechanics phenomena to perform computation, such that:
  - Qubits, particles that represent the information, can exist in both state 0 and 1 simultaneously (*superposition*).
  - Two qubits may be somehow correlated and described as a single object with four states, even if physically distant (*entanglement*).

- Operations are described using circuits: a sequence of quantum logic gates applied to qubits. A N-qubits quantum computer can process 2^N operations in parallel.

- Public-key cryptography is used to:
  - Establish a shared secret key (e.g. Diffie-Hellman key exchange).
  - Authenticate one another via digital signatures (e.g. ECDSA).

  Symmetric algorithms use a secret key to encrypt stuff (e.g. AES).

Introduction

- Post-quantum cryptography is cryptography under the assumption that the attacker has a strong quantum computer; such cryptosystems are intended to remain secure even in this scenario.

- As of now, it is still unclear whether a large quantum computer will be technically realizable one day, and yet, if quantum computing scales as expected, then RSA/DH/ECC will all be broken.

- Thus, while the advent of quantum computing has the potential to facilitate unprecedented scientific advances, it exposes today's strongest encryption algorithms to a serious threat.

Introduction

- Currently, all public-key cryptography algorithms used in practice today are built on number-theoretic problems, which can be efficiently solved on quantum computers, as shown by Shor in 1994.

- To withstand quantum attacks, NIST, in 2015, initiated the process of standardizing post-quantum public-key encryption schemes, key encapsulation mechanisms, and digital signature algorithms.
  - $\rightarrow$ In February 2016, NIST officially announced calls for quantum-safe submissions. 26 algorithms passed the analysis so far, the last round of evaluation is scheduled to be held in 2020.

- Post-quantum algorithms can be categorized into different classes, including code-based, lattice-based, hash-based, isogeny-based, and multivariate cryptography.

Shor's algorithm

- The most efficient method to break integer factorization, which underpins RSA, runs in superpolynomial time on a classic computer.
  - $\rightarrow$ A variant of Shor's uses $\mathcal{O}(n^3 \log n)$ ops on $2n+3$ qubits, with n size of $N = pq$, thus providing an exponential speedup.

- The general idea behind is to reduce the factorization of N to the task of finding the period of a modular exponentiation function $\mathcal{F}(a) = x^a \ mod \ N$, where x is picked up randomly, coprime to N.
  1. Being $\mathcal{F}(a)$ periodic, then it has some order r such that $x^r \equiv 1 \ mod \ N$. If r found is even, then $(x^{r/2} - 1)(x^{r/2} + 1) \equiv_N 0$. Thus, at least one of $(x^{r/2} \pm 1)$ is a divisor of N.
  2. To find period r, evaluate $\mathcal{F}(a)$ on a superposition of all inputs a, which is done only once due to quantum parallelism.
  3. Apply quantum Fourier transform to compare all the calculated values and obtain - after measurement - the period of the function, which reveals a factor of N with high probability.

Grover's algorithm

- Some symmetric cryptographic schemes can be affected by Grover's algorithm, presented in 1996.
- The algorithm can search an unsorted database of N entries using $\sqrt{N}$ quantum queries (instead of classic $N/2$ searches).
- It can be utilized to brute-force AES in $\mathcal{O}(2^{n/2})$ time, leading to 2^{64} for 128 bit keys. Likewise, it can be used to find collisions in some hash functions in combination with birthday paradox.
- Although Grover's speedup ($N \to \sqrt{N}$) is not as dramatic as Shor's one, it is recommended, at least, to double the key length, i.e. switching to 256-bit AES keys and using new SHA-2/SHA-3 hash family.

Computational complexity

- A new quantum complexity class is introduced: *Bounded-error Quantum Polynomial time* (BQP), the class of decision problems feasible for a quantum computer, which is suspected to be disjoint from NP-complete and a strict superset of P.
  - $\rightarrow$ This is why we are looking for NP-hard one-way functions.

- Integer factorization and discrete logarithm are in BQP.
  - $\rightarrow$ Indeed, quantum computing can be used to solve efficiently some NP problems.

- Quantum algorithms are *probabilistic*, i.e. the result may be returned with some small error. By running it repeatedly, the accepted answer will be the one that appears more than 2/3 of the times.

Code-based encryption

Code-based encryption

- Coding theory is used to ensure that information is transmitted accurately over a noisy channel.

- Additional bits (*error-correcting codes*) can be attached to each message so that errors (flipped bits) may be detected and corrected.
  - $\rightarrow$ Been extensively used in a lot of applications, including hard disks, satellite broadcasting, and fault-tolerant quantum computation.

- Typically, k bits of data gets stored in n bits ($n > k$), adding some extra bits.

- In 1978, McEliece proposed using a generator matrix as a public key and encrypting a codeword by adding a specified number of errors to it.

Linear codes

A binary linear code $C\,[n, k]$ of length n and dimension k, is a k-dimensional subspace of the vector space $\mathbb{F}_2^n$ ($n - k$ is the redundancy). It can be represented by a $k \times n$ *generator matrix* G such that $C = \{\boldsymbol{mG} \mid \boldsymbol{m} \in \mathbb{F}_2^k\}$, or by a *parity-check matrix* H such that $C = \{\boldsymbol{c} \in \mathbb{F}_2^n \mid \boldsymbol{Hc}^T = 0\}$.

Example: equations below are satisfied if no error occurred with parity check matrix ($n = 7$, $k = 4$) and bit string $b = (b_0, b_1, b_2, b_3, b_4, b_5, b_6)$.

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{cases} b_0 + b_1 + b_3 + b_4 = 0 \\ b_0 + b_2 + b_3 + b_5 = 0 \\ b_1 + b_2 + b_3 + b_6 = 0 \end{cases}$$

The codewords are obtained by multiplying the vectors from $\mathbb{F}_2^k$ with the row space of G, i.e. $mG = c$ (still a vector!), thus having a total of 2^k codewords.

The *weight* of a vector is the number of non-zero it has, the *minimum distance* of a code C is the smallest weight of any nonzero codeword in C.

McEliece PKE

- McEliece public-key encryption scheme works as follow:
  1. Choose a linear code $\mathcal{C}$ $[n, k]$ with generator matrix $G_{k \times n}$, able to correct up to t errors.
  2. Compute $k \times n$ matrix $\hat{G} = SGP$, where $S_{k \times k}$ (*scrambling*) and $P_{n \times n}$ (*permutation*) are two randomly selected invertible matrices so that the inner structure of G is perturbed. $K_{pub} = \hat{G}$, $K_{priv} = \langle S, G, P \rangle$.
  3. The ciphertext is computed as $c = m\hat{G} + e$, where $m \in \mathbb{F}_2^k$ are messages suitable for encryption and e is a random n-bit error vector with weight t.

- Lack of knowledge on how to perform the inverse transformation leads the attacker to face a presumably hard problem:

$$c = \overbrace{m\hat{G} + e}^{\text{hard}} = m(SGP) + e$$

McEliece PKE

- Hard problem: find the closest codeword $m\hat{G} \in \mathcal{C}$ (assuming it is unique) to a given $c \in \mathbb{F}_2^k$, where $m\hat{G}$ is t-errors away from c.
  - $\rightarrow$ Decoding a random-looking code up to its error-correcting capability has been proved to be intractable.

- Original construction is built over *Goppa codes* which have a fast decoding algorithm and remain still unbroken (unlike other family of codes which turned out to be insecure).
  - $\rightarrow$ Hence, McEliece is secure as long as there is no algorithm that distinguishes between binary Goppa codes and random binary codes.

- Knowing the structure of the underlying linear code (secret key), by applying an efficient decoding algorithm $\mathcal{D}$ able to remove t errors present in cP^{-1}, the decryption is relatively easy:

$$\mathcal{D}(cP^{-1})S^{-1} = \mathcal{D}(mSG + eP^{-1})S^{-1} = (mS)S^{-1} = m$$

McEliece PKE

- Quantum Fourier sampling attacks do not apply to McEliece.[1] The best known attacks against McEliece's system are based on *information-set decoding*, but still takes exponential time.
- Chosen-ciphertext attacks can be prevented by encrypting and hashing a random codeword, and use this one as secret key for successive encryptions.
- Encryption and decryption are much faster than RSA (multiplication of a vector by a matrix vs raising a number to a power).
- The main obstacle to adoption is the key size (too large matrices), which can reach 1 MB for a high degree of security.

[1] *The McEliece Cryptosystem Resists Quantum Fourier Sampling Attacks* [Dinh et al, 2010].

Lattice-based cryptography

Lattice-based cryptography

- Despite being a rather new research area, lattice-based cryptography gained interest - due to its high efficiency - to become one of the most promising candidate for quantum-resistant cryptography.
- A *lattice* is a set of points in an n-dimensional space with a periodic structure. Formally, given the *basis* vectors $\boldsymbol{b_1}, \ldots, \boldsymbol{b_n} \in \mathbb{Z}^n$, the set of the integer linear combinations of all of them creates a lattice:

$$L = \left\{ \sum_{i=1}^{n} b_i \cdot x_i \mid x_i \in \mathbb{Z}, b_i \in \mathbb{Z}^n \right\}$$

- One of the most important hard problem is the *shortest vector problem*, which consists of finding the shortest nonzero vector in a lattice given as input an arbitrary basis for it.
  - $\rightarrow$ The higher the dimension of the lattice, the more infeasible the problem. Cryptosystems involving lattices are built on the inherent intractability of solving this kind of problem.

Lattice-based cryptography

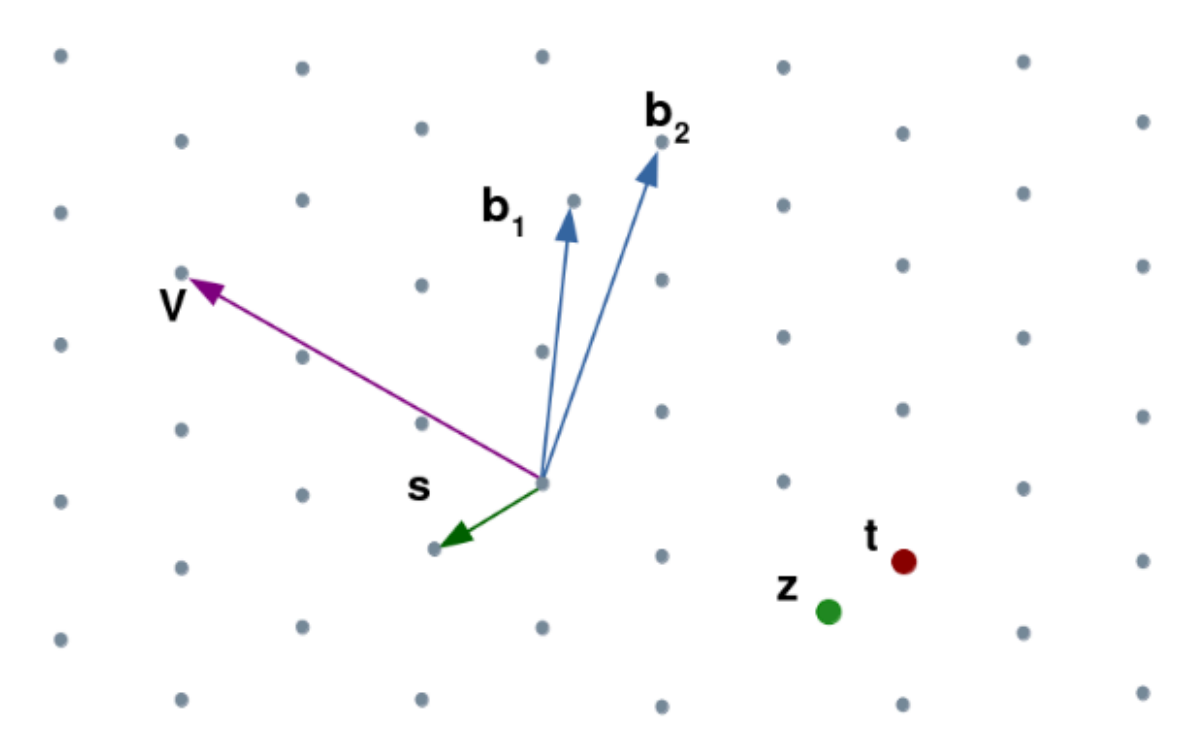

Figure 1: Solving the SVP problem involves finding the shortest nonzero vector in a lattice. In the graph, the vector s is the shortest one. The problem gets harder as the dimensions increase.

Lattice-based cryptography

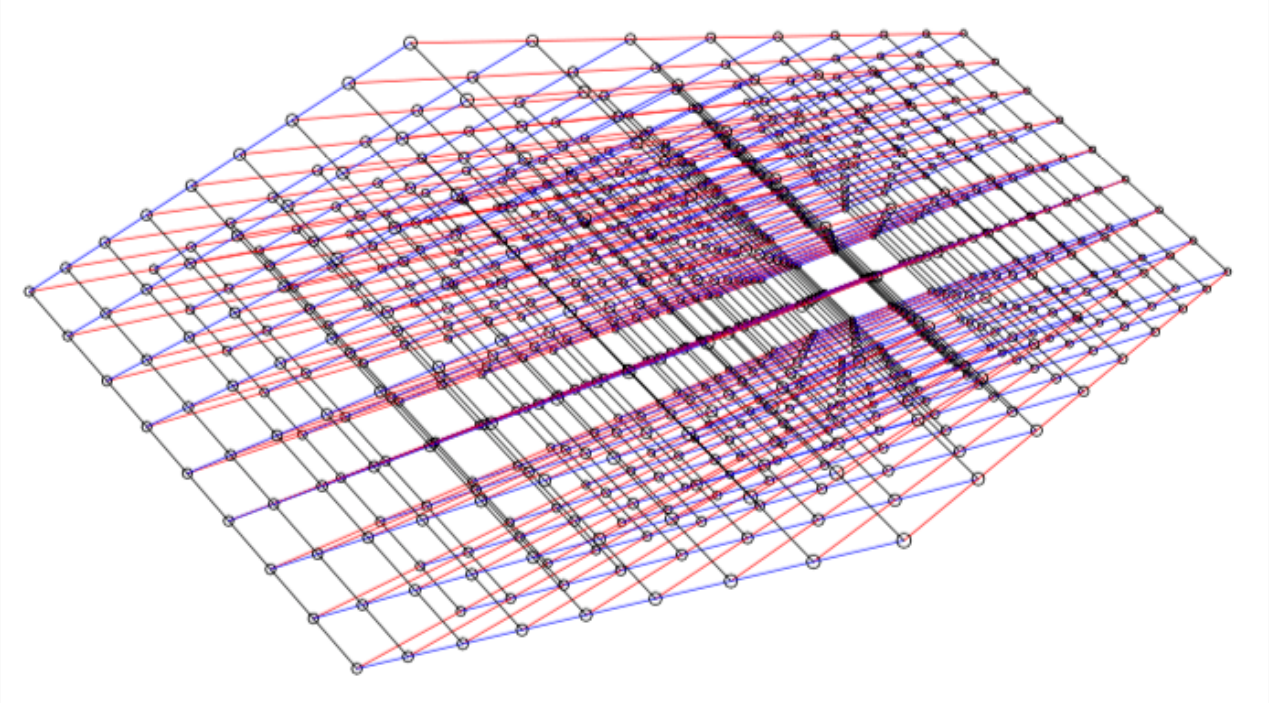

Figure 2: View of a $9 \times 9 \times 9$ subset of a non-orthogonal 3-dimensional lattice. Lattice-based cryptography hides a point in a high-dimensional lattice *mod*q by making small changes to all coordinates (*closest vector problem*).

NTRUEncrypt

- NTRU is a very fast public-key cryptosystem whose underlying problem is based on the SVP in lattices. Operations are done in the truncated polynomial ring $R = \mathbb{Z}[X]/(X^p - 1)$ *mod* q (p, q coprime).
  - Key generation: let $f, g \in R$ be private with coefficients $\{-1, 0, 1\}$ (the smaller the higher the security), the public key is a p-coefficient polynomial:

    $$h \equiv pf^{-1} \times g \; mod \; q \equiv_q h_0 + h_1 x + \ldots + h_{p-1} x^{p-1}$$

    where $\times$ is the convolution product and f^{-1} computed w/ EEA.
  - Encryption: be $m \in R$ a message in form of a polynomial with coefficients in $\{-1, 0, 1\}$, pick a random polynomial $r \in R$, the ciphertext is computed as:

    $$c \equiv r \times h + m \; mod \; q \; (mod \; x^{p-1})$$

  - Decryption: the receiver computes $a \equiv_q f \times c$, transforms polynomial a so that coefficients are in $[-q/2, q/2]$ and finds $m \equiv f^{-1} \times a \; mod \; p$.

NTRUEncrypt

- Define L as the set of pairs (u, v) of p-coefficient polynomial such that $hu - v \bmod q \bmod x^p - 1 = 0$. Then L is a lattice in $2p$-dimensional space and contains a point close to $(0, c)$, i.e. $(r, c - e)$.
  $\rightarrow$ Attacker needs to find a lattice point close to a given point.

- Choosing wrong parameters may lead to decryption failure and private key recover, so attention must be paid.

- The quasi-cyclotomic structure of $x^p - 1$ has been used to break other lattice-based algorithms. Although there are no known attacks exploiting this structure in NTRU, it is recommended to do math over $x^p - x - 1$.

Lattice-based signatures

- Conversely, original NTRUSign protocol is considered to be broken as signatures leak information about the private key.
- The most interesting lattice-based signature systems are based on Lyubashevsky's scheme from 2012, whose security relies on the generalization of the Knapsack problem to arbitrary rings.
  - The secret key is a matrix $S \in \mathbb{Z}_q^{m \times k}$ with entries restricted to $\{-1, 0, 1\}$, the public key consists of the matrices $A \in \mathbb{Z}_q^{n \times m}$ and $T = AS \bmod q$.
  - The signer picks $y \in \mathbb{Z}_q^m$ from an m-dimensional distribution (like the Gaussian one); then computes $c = H(Ay \bmod q, \mu)$ (where μ is the message) and $z = Sc + y$. The signature is the pair $\langle c, z \rangle$.
  - To verify the signature, c and z must be sufficiently small and $H(Az - Tc \bmod q, \mu)$ equals c.

Multivariate-quadratic-equation signatures

Multivariate-quadratic-equation cryptography

- Solving systems of multivariate polynomials equations over a finite field GF is proven to be NP-hard.

- As a comparison, encryption in RSA requires a modular exponentiation ($y = x^e \, (mod \; n)$), while encryption in a MQ-problem with 5 unknowns and 5 equations consists of solving (for instance):

$$\mathcal{P} = \begin{cases} y_1 = x_1^2 + x_3 x_4 + x_5 + 1 \\ y_2 = x_3^2 + 2x_2 x_5 + x_1 + x_2 + 2 \\ \vdots \\ y_5 = x_4^2 + 4x_1 x_5 + 2x_3 x_2 + x_1 + 3 \end{cases}$$

- Under $GF(2)$ and $m \approx n$, the best known algorithm is exhaustive search twice (which has a complexity of $\mathcal{O}(2^{2n})$).

HFE^{v-}

- Introduced in 1996, a well-known multivariate signature scheme is the HFE^{v-} (slight different variation of the Hidden Field Equations), which is characterized by its very short signatures.
  - The public key is a sequence of polynomials $p_1, \ldots, p_m$ each with n variables over $GF(2)$.
  - The signer chooses the polynomials with a secret structure such that he/she can solve $p_m(s_1, \ldots, s_n) = h_m$ (m equations), and so on w/ $(h_1, \ldots, h_m)$ the hash of the message.
  - Private affine transformations (two secret invertible matrices, S and T) are used to hide the private polynomials.
  - Without the trapdoor (*secret polys*, S, T), it is computationally infeasible to obtain a solution for the system of equations.

- Intuitively, solving such systems requires knowing *how* they have been constructed (which is the secret key). For HFE^{v-} in particular, all known attacks take exponential time.

Hash-based signatures

Hash-based signatures

- A hash-based digital signature scheme leverages the security assumptions of the underlying hash function (finding the preimage for a given output string is hard!).

- Lamport's one-time signature works as follows:
  - Hash a message M_i (w/ e.g. SHA-256) and choose randomly 512 bit strings. Then $(x_0, y_0, \ldots, x_{255}, y_{255})$ will be the private key, while $(H(x_0), H(y_0), \ldots, H(x_{255}), H(y_{255}))$ will be the public key. To sign a message, you reveal the secret x_i or y_i (depending if $H(M_i)$ is a 0 or 1) for each bit of the hash of message. The signature is literally half of the private key.

- Merkle's signature extends this scheme by combining 2^k public keys into one which can then be used to verify all 2^k signatures.

Takeaways

- While currently only small-scale quantum computers exist, it is not implausible that a large-scale quantum computer which can break current PKC can be built within the next one or two decades.

- There is the need to act now!
  - $\rightarrow$ Deploying a cryptographic system incurs in non-negligible physical costs (time/energy consumed due to the computation).
  - $\rightarrow$ Update to a new cryptographic infrastructure is complex (different HW accelerators, custom SW, ensure compatibility...); hence the transition can take a very long time.
  - $\rightarrow$ Still a lot of research is required to analyze all the attacker capabilities (especially cache and time side-channel attacks).

To follow up

1. D. Bernstein, T. Lange, *Post-quantum cryptography – dealing with the fallout of physics success*, https://eprint.iacr.org/2017/314.pdf.
2. V. Mavroeidis, K. Vishi et al, *The Impact of Quantum Computing on Present Cryptography*, https://arxiv.org/pdf/1804.00200.pdf.
3. S. Au, C. Turner, J.Everson, *The McEliece Cryptosystem*, http://www.math.unl.edu/~s-jeverso2/McElieceProject.pdf.
4. Z. Zhang, *NTRUEncrypt and pqNTRUSign*, https://csrc.nist.gov/CSRC/ntruencrypt-pqntrusign.pdf
5. M. Baldi, A. Barenghi et al, *LEDACrypt*, https://csrc.nist.gov/CSRC/ledacrypt.pdf.
6. C. Costello, *An introduction to supersingular isogeny-based cryptography*, https://ecc2017.cs.ru.nl/slides/ecc2017costello.pdf.

Thanks!