Introduction à la cryptographie post-quantique

Jean-Christophe Deneuville

<jean-christophe.deneuville@enac.fr>

November 30, 2023

Outline

Disclaimer !!

(dans l'éventualité où nous n'aurions pas le temps)

Disclaimer

What are the alternatives to classical cryptography in presence of an adversary equipped with a large scale quantum computer?

Disclaimer

What are the alternatives to classical cryptography in presence of an adversary equipped with a large scale quantum computer?

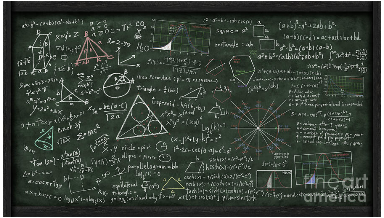
- Quantum Key Exchange (out of the scope of this talk)

Disclaimer

What are the alternatives to classical cryptography in presence of an adversary equipped with a large scale quantum computer?

- Quantum Key Exchange (out of the scope of this talk)

- Post-Quantum Cryptography



Recalls on hash functions

A hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$:

Recalls on hash functions

A hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$:

- takes arbitrarily long bit string input

Recalls on hash functions

A hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^{\ell}$:

- takes arbitrarily long bit string input
- outputs fix-length bit strings

Recalls on hash functions

A hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$:

- takes arbitrarily long bit string input
- outputs fix-length bit strings
- is deterministic

Recalls on hash functions

A hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$:

- takes arbitrarily long bit string input
- outputs fix-length bit strings
- is deterministic
- is pseudo-random (with avalanche effect)

Recalls on hash functions

A hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$:

- takes arbitrarily long bit string input
- outputs fix-length bit strings
- is deterministic
- is pseudo-random (with avalanche effect)

Some examples: MD5, SHA-1, SHA-2, SHA-3, SHA-256, BSD, CRC, MD6, ...

Recalls on hash functions

A hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$:

- takes arbitrarily long bit string input
- outputs fix-length bit strings
- is deterministic
- is pseudo-random (with avalanche effect)

Some examples: MD5, SHA-1, SHA-2, SHA-3, SHA-256, BSD, CRC, MD6, ...

Demo with SHA-1 (OpenSSL).

Hash functions properties

The main properties of a *secure* hash function $\mathcal{H}$ are:

Hash functions properties

The main properties of a *secure* hash function $\mathcal{H}$ are:

- Collision resistance :
  It is computationally infeasible to find a couple $x \neq y \in \{0,1\}^*$ such that $\mathcal{H}(x) = \mathcal{H}(y)$.

Hash functions properties

The main properties of a *secure* hash function $\mathcal{H}$ are:

- Collision resistance :
  It is computationally infeasible to find a couple $x \neq y \in \{0,1\}^*$ such that $\mathcal{H}(x) = \mathcal{H}(y)$.
- First pre-image resistance :
  Given $h = \mathcal{H}(x)$ for some (unknown) x, it is computationally infeasible to find a y such that $\mathcal{H}(y) = h$.

Hash functions properties

The main properties of a *secure* hash function $\mathcal{H}$ are:

- Collision resistance :
  It is computationally infeasible to find a couple $x \neq y \in \{0,1\}^*$ such that $\mathcal{H}(x) = \mathcal{H}(y)$.
- First pre-image resistance :
  Given $h = \mathcal{H}(x)$ for some (unknown) x, it is computationally infeasible to find a y such that $\mathcal{H}(y) = h$.
- Resistance to the second pre-image :
  Given x (this time x is known), it is computationally infeasible to find a $y \neq x$ such that $\mathcal{H}(y) = \mathcal{H}(x)$.

Security of hash functions

Security of a hash function (collision-resistance), birthdays' paradox

Security of hash functions

Security of a hash function (collision-resistance), birthdays' paradox

Question

How many people do you need to gather at least to have more than a one in two chance of having two born on the same day?

Security of hash functions

Security of a hash function (collision-resistance), birthdays' paradox

Question

How many people do you need to gather at least to have more than a one in two chance of having two born on the same day?

Answer

23 people for more than one chance in two. The odds increase to $\sim 90\%$ with only 41 people.

(the magic of the exponential!)

Security of hash functions

23 people: explanation

We're not looking for two people who were born on a certain day (*e.g.* your birthday), but any day!

Security of hash functions

23 people: explanation

We're not looking for two people who were born on a certain day (*e.g.* your birthday), but any day!

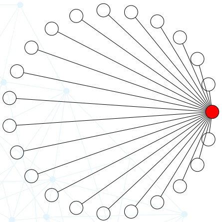
Probability $\mathbb{P}$ that there is no collision :

Security of hash functions

23 people: explanation

We're not looking for two people who were born on a certain day (*e.g.* your birthday), but any day!

Probability $\mathbb{P}$ that there is no collision :

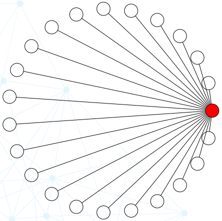


Security of hash functions

23 people: explanation

We're not looking for two people who were born on a certain day (*e.g.* your birthday), but any day!

Probability $\mathbb{P}$ that there is no collision :



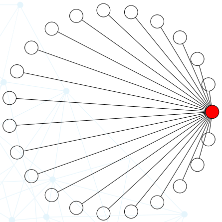
$$\mathbb{P} = \frac{365-23}{365} = 93.70\%$$

Security of hash functions

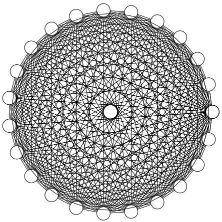
23 people: explanation

We're not looking for two people who were born on a certain day (*e.g.* your birthday), but any day!

Probability $\mathbb{P}$ that there is no collision :



$$\mathbb{P} = \frac{365 - 23}{365} = 93.70\%$$

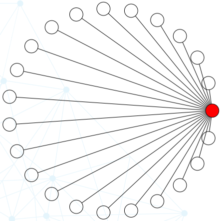


Security of hash functions

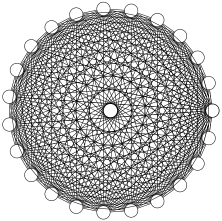
23 people: explanation

We're not looking for two people who were born on a certain day (*e.g.* your birthday), but any day!

Probability $\mathbb{P}$ that there is no collision :



$$\mathbb{P} = \frac{365 - 23}{365} = 93.70\%$$



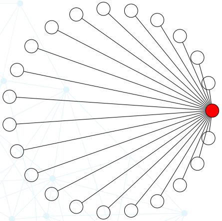
$$\mathbb{P} = \prod_{i=1}^{23} \frac{365 - i}{365} = 49.27\%$$

Security of hash functions

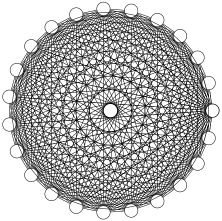
23 people: explanation

We're not looking for two people who were born on a certain day (*e.g.* your birthday), but any day!

Probability $\mathbb{P}$ that there is no collision :



$$\mathbb{P} = \frac{365-23}{365} = 93.70\%$$



$$\mathbb{P} = \prod_{i=1}^{23} \frac{365-i}{365} = 49.27\%$$

Demo (real if sufficiently many, simulation otherwise)

Security of hash functions

Application

$\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$. How many elements of $\{0,1\}^*$ will we have to sample before finding two elements with the same hash (*i.e.* a collision) with a good probability?

Security of hash functions

Application

$\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$. How many elements of $\{0,1\}^*$ will we have to sample before finding two elements with the same hash (*i.e.* a collision) with a good probability?

Only $2^{n/2}$!

Security of hash functions

Application

$\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$. How many elements of $\{0,1\}^*$ will we have to sample before finding two elements with the same hash (*i.e.* a collision) with a good probability?

Only $2^{n/2}$!

... but what does it mean? How hard is 2^{100} for instance?

Computing power / security level

Computing power in 2020

Computing power / security level

Computing power in 2020

For a standard machine: 64-bit architecture

2^6

Computing power / security level

Computing power in 2020

For a standard machine: 16 cores

$2^6 \times 2^4$

Computing power / security level

Computing power in 2020

For a standard machine: 4 GHz

$2^6 \times 2^4 \times 2^2 \times 10^9$

Computing power / security level

Computing power in 2020

Let it run for a year

$$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365$$

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4$$

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4 \approx 2^{80}$ elementary operations

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4 \approx 2^{80}$ elementary operations

This simple calculation does not take into account the computing power offered by GPUs, ASICS, ...

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4 \approx 2^{80}$ elementary operations

This simple calculation does not take into account the computing power offered by GPUs, ASICS, ...

"Theorem" 2020 (Disclaimer: This theorem will not be valid forever)

In 2020, it is estimated that anything that requires a computing power greater than 2^{100} elementary operations is infeasible.

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4 \approx 2^{80}$ elementary operations

This simple calculation does not take into account the computing power offered by GPUs, ASICS, ...

"Theorem" 2020 (Disclaimer: This theorem will not be valid forever)

In 2020, it is estimated that anything that requires a computing power greater than 2^{100} elementary operations is infeasible.

A relatively concrete example

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4 \approx 2^{80}$ elementary operations

This simple calculation does not take into account the computing power offered by GPUs, ASICS, ...

"Theorem" 2020 (Disclaimer: This theorem will not be valid forever)

In 2020, it is estimated that anything that requires a computing power greater than 2^{100} elementary operations is infeasible.

A relatively concrete example

Globally

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4 \approx 2^{80}$ elementary operations

This simple calculation does not take into account the computing power offered by GPUs, ASICS, ...

"Theorem" 2020 **(Disclaimer: This theorem will not be valid forever)**

In 2020, it is estimated that anything that requires a computing power greater than 2^{100} elementary operations is infeasible.

A relatively concrete example

Globally, and with a substantial investment

Computing power / security level

Computing power in 2020

Imagine that a government agency can easily acquire 10,000 machines

$2^6 \times 2^4 \times 2^2 \times 10^9 \times 60 \times 60 \times 24 \times 365 \times 10^4 \approx 2^{80}$ elementary operations

This simple calculation does not take into account the computing power offered by GPUs, ASICS, ...

"Theorem" 2020 **(Disclaimer: This theorem will not be valid forever)**

In 2020, it is estimated that anything that requires a computing power greater than 2^{100} elementary operations is infeasible.

A relatively concrete example

Globally, and with a substantial investment, there was 2^{89} hash SHA-256 made on the blockchain BitCoin in 2019...

Cryptographic protocol recall/presentation

Corollary

In 2020, for short-term security, it is recommended to use parameters providing a security level equivalent to 128 bits.

Cryptographic protocol recall/presentation

Corollary

In 2020, for short-term security, it is recommended to use parameters providing a security level equivalent to 128 bits.

Paranoid, you'll become...

In 2020, prefer algorithms offering a security level equivalent to 256 bits.

Cryptographic protocol recall/presentation

Corollary

In 2020, for short-term security, it is recommended to use parameters providing a security level equivalent to 128 bits.

Paranoid, you'll become...

In 2020, prefer algorithms offering a security level equivalent to 256 bits.

Deprecated!

This increase in computing power makes some "old" algorithms obsolete, such as:

- DES (64 secret key bits, 56 only for security)
- MD≤ 5 (128 bits output for MD5 (paradox + weaknesses))
- SHA-≤ 1 (160 bits output)...

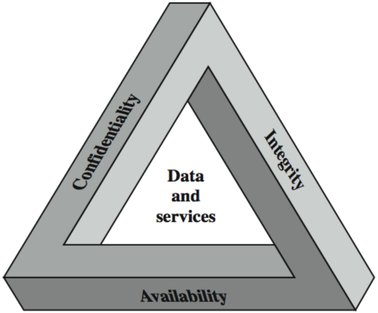
Outline

Outline

Sécurité informatique (security)

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Définition du NIST:
The protection afforded to an automated information system in order to attain the applicable objectives of preserving the **integrity**, **availability** and **confidentiality** of information system resources (includes hardware, software, firmware, information/data, and telecommunications)



http://csrc.nist.gov/publications/fips/fips199/FIPS-PUB-199-final.pdf

Sécurité informatique (security)

3 piliers fondamentaux de la security

- **Confidentialité**
  → Restrictions concernant l'**accès** aux informations et leur **divulgation** aux **seules personnes autorisées**, y compris les moyens de protéger la **vie privée** et les informations exclusives.
- **Intégrité**
  → Protection contre la **modification** ou la **destruction** des informations, y compris la garantie de la **non-répudiation** et de l'**authenticité** des informations.
- **Disponibilité**
  → Garantir un **accès rapide** et **fiable** aux informations et leur utilisation.

Outline

Main objectives of Cryptography

Cryptography embodies the theoretical aspects of *security*.

Main objectives of Cryptography

Cryptography embodies the theoretical aspects of *security*.

Context: 

wishes to send message to 

Main objectives of Cryptography

Cryptography embodies the theoretical aspects of *security*.

Context: 

wishes to send message  to 

Cryptography goal, to guarantee:

1. Authentication
2. Confidentiality
3. Integrity
4. Non-Répudiation

Main objectives of Cryptography

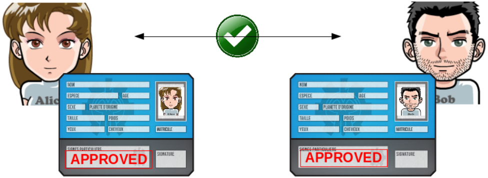
Cryptography embodies the theoretical aspects of *security*.

Context: 

wishes to send message  to 

Cryptography goal, to guarantee:

1. Authentication
2. Confidentiality
3. Integrity
4. Non-Répudiation



Main objectives of Cryptography

Cryptography embodies the theoretical aspects of *security*.

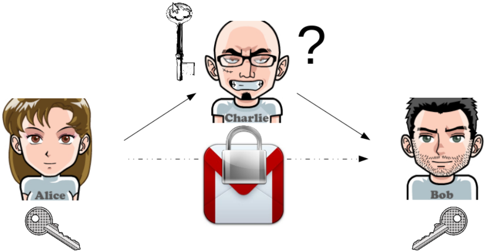
Context: 

wishes to send message 
to 

Cryptography goal, to guarantee:

1. Authentication 

2. Confidentiality
3. Integrity
4. Non-Répudiation



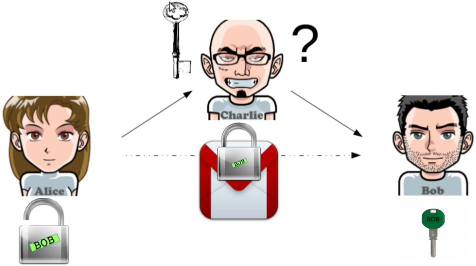
Main objectives of Cryptography

Cryptography embodies the theoretical aspects of *security*.

Context:  wishes to send message  to 

Cryptography goal, to guarantee:

1. Authentication 
2. Confidentiality
3. Integrity
4. Non-Répudiation



Main objectives of Cryptography

Cryptography embodies the theoretical aspects of *security*.

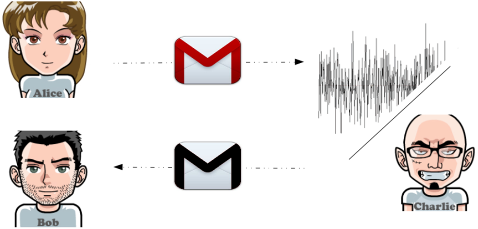
Context: 

wishes to send message  to 

Cryptography goal, to guarantee:

1. Authentication 

2. Confidentiality 
3. Integrity
4. Non-Répudiation



Main objectives of Cryptography

Cryptography embodies the theoretical aspects of *security*.

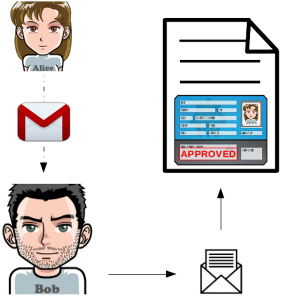
Context: 

wishes to send message  to 

Cryptography goal, to guarantee:

1. Authentication 

2. Confidentiality 
3. Integrity 
4. Non-Répudiation

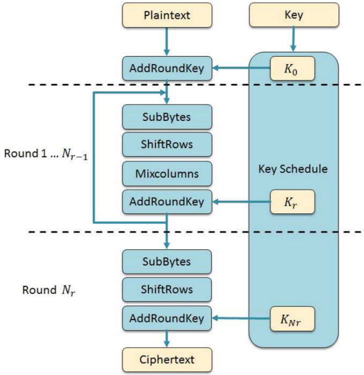


Outline

Outline

Crypto symétrique : AES

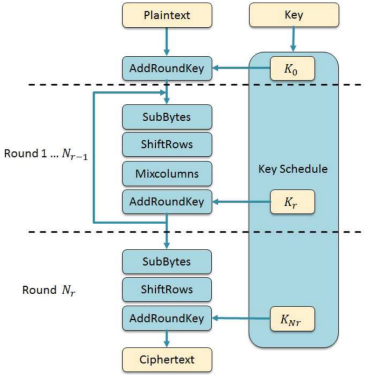
Advanced **E**ncryption **S**tandard



Source: S. Ordas

Crypto symétrique : AES

Advanced Encryption Standard

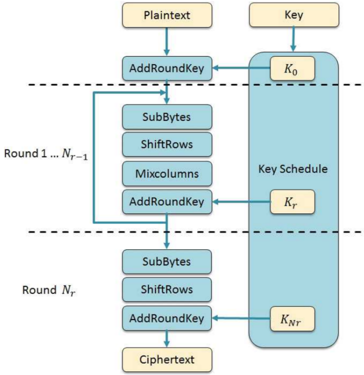


Source: S. Ordas

- Symmetric encryption standard since 2000

Crypto symétrique : AES

Advanced **E**ncryption **S**tandard

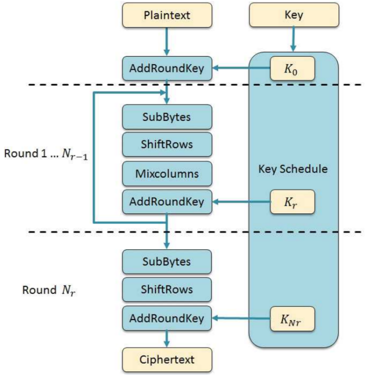


Source: S. Ordas

- Symmetric encryption standard since 2000
- Supports 128, 192 and 256-bit keys

Crypto symétrique : AES

Advanced **E**ncryption **S**tandard

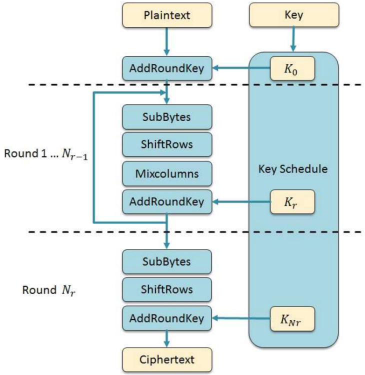


Source: S. Ordas

- Symmetric encryption standard since 2000
- Supports 128, 192 and 256-bit keys
- Respectively on 10, 12 and 14 rounds

Crypto symétrique : AES

Advanced **E**ncryption **S**tandard

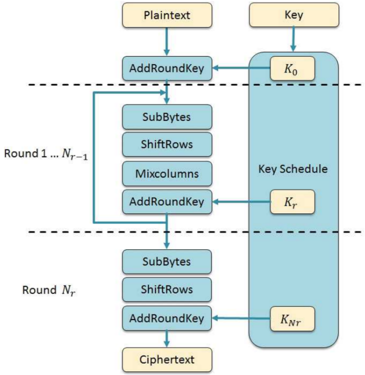


Source: S. Ordas

- Symmetric encryption standard since 2000
- Supports 128, 192 and 256-bit keys
- Respectively on 10, 12 and 14 rounds
- The key is first extended into as many round keys as necessary

Crypto symétrique : AES

Advanced **E**ncryption **S**tandard



- Symmetric encryption standard since 2000
- Supports 128, 192 and 256-bit keys
- Respectively on 10, 12 and 14 rounds
- The key is first extended into as many round keys as necessary
- Best known attacks only marginally affect security

Source: S. Ordas

Crypto asymétrique : New directions in cryptography

In 1976, Diffie and Hellman proposed a protocol for exchanging keys remotely and securely [DH76] **(or almost)**.

Crypto asymétrique : New directions in cryptography

In 1976, Diffie and Hellman proposed a protocol for exchanging keys remotely and securely [DH76] **(or almost)**.

Public Settings :

- p a (large) prime number
- g a generator of the cyclic group $(\mathbb{Z}/p\mathbb{Z})^*$

Crypto asymétrique : New directions in cryptography

In 1976, Diffie and Hellman proposed a protocol for exchanging keys remotely and securely [DH76] **(or almost)**.

Public Settings :

- p a (large) prime number
- g a generator of the cyclic group $(\mathbb{Z}/p\mathbb{Z})^*$

p, g

Crypto asymétrique : New directions in cryptography

In 1976, Diffie and Hellman proposed a protocol for exchanging keys remotely and securely [DH76] (or almost).

Public Settings :

- p a (large) prime number
- g a generator of the cyclic group $(\mathbb{Z}/p\mathbb{Z})^*$

p, g

$a \in (\mathbb{Z}/p\mathbb{Z})^*, A = g^a \mod p$

$b \in (\mathbb{Z}/p\mathbb{Z})^*, B = g^b \mod p$

Crypto asymétrique : New directions in cryptography

In 1976, Diffie and Hellman proposed a protocol for exchanging keys remotely and securely [DH76] (or almost).

Public Settings :

- p a (large) prime number
- g a generator of the cyclic group $(\mathbb{Z}/p\mathbb{Z})^*$

p, g

$a \in (\mathbb{Z}/p\mathbb{Z})^*, A = g^a \mod p$

$b \in (\mathbb{Z}/p\mathbb{Z})^*, B = g^b \mod p$

$$\xrightarrow{\quad A \quad}$$

$$\xleftarrow{\quad B \quad}$$

Crypto asymétrique : New directions in cryptography

In 1976, Diffie and Hellman proposed a protocol for exchanging keys remotely and securely [DH76] (or almost).

Public Settings :

- p a (large) prime number
- g a generator of the cyclic group $(\mathbb{Z}/p\mathbb{Z})^*$

p, g

$a \in (\mathbb{Z}/p\mathbb{Z})^*, A = g^a \mod p$

$b \in (\mathbb{Z}/p\mathbb{Z})^*, B = g^b \mod p$

$\xrightarrow{A}$

$\xleftarrow{B}$

$B^a = (g^b)^a = g^{ab} \mod p$

$A^b = (g^a)^b = g^{ab} \mod p$

Premier schéma asymétrique : RSA

Two years later, Rivest, Shamir and Adleman published the first asymmetric encryption scheme [RSA78].

Premier schéma asymétrique : RSA

Two years later, Rivest, Shamir and Adleman published the first asymmetric encryption scheme [RSA78].

p, q large primes, $N = pq$

e co-prime with $\varphi(N) = (p-1)(q-1)$

$d = e^{-1} \mod \varphi(N)$

$(m^e)^d \mod N = m$

message $m \in \mathbb{Z}/N\mathbb{Z}$

N, e →

← $m^e \mod N$

$m^e \mod N$

Premier schéma asymétrique : RSA

Two years later, Rivest, Shamir and Adleman published the first asymmetric encryption scheme [RSA78].

p, q large primes, $N = pq$

e co-prime with $\varphi(N) = (p-1)(q-1)$

message $m \in \mathbb{Z}/N\mathbb{Z}$

$$d = e^{-1} \mod \varphi(N)$$

N, e $\longrightarrow$

$m^e \mod N$ $\longleftarrow$

$$m^e \mod N$$

$$(m^e)^d \mod N = m$$

Fermat's (little) theorem

If p is prime, then $\forall a \in \mathbb{Z}/p\mathbb{Z}$, it holds that : $a^{p-1} = 1 \mod p$.

Cryptanalyse de ces standards sur une architecture classique (1/2)

Problèmes mathématiques sous-jacents :

- problème RSA : *étant donnés* N, c, *et* e, *trouver* m *tel que* $m^e = c \mod N$
- problème DH : *étant donnés* p, g, $g^a \mod p$ *et* $g^b \mod p$, *trouver* $g^{ab} \mod p$

Cryptanalyse de ces standards sur une architecture classique (1/2)

Problèmes mathématiques sous-jacents :

- problème RSA : *étant donnés* N, c, *et* e, *trouver* m *tel que* $m^e = c \mod N$
- problème DH : *étant donnés* p, g, $g^a \mod p$ *et* $g^b \mod p$, *trouver* $g^{ab} \mod p$

Problèmes connexes :

- FACT : *étant donné* $N = pq$ *avec* p *et* q *premiers distincts, trouver* p *et* q
- DL : *étant donné* p *premier,* g *et* $g^x \mod p$, *trouver* x

Cryptanalyse de ces standards sur une architecture classique (1/2)

Problèmes mathématiques sous-jacents :

- problème RSA : *étant donnés* N, c, *et* e, *trouver* m *tel que* $m^e = c \mod N$
- problème DH : *étant donnés* p, g, $g^a \mod p$ *et* $g^b \mod p$, *trouver* $g^{ab} \mod p$

Problèmes connexes :

- FACT : *étant donné* $N = pq$ *avec* p *et* q *premiers distincts, trouver* p *et* q
- DL : *étant donné* p *premier,* g *et* $g^x \mod p$, *trouver* x

Lien entre ces problèmes :

- Si FACT est facile, Alors RSA est facile *(comprendre RSA cassé)*
- Si DL est facile, Alors DH est facile

Cryptanalyse de ces standards sur une architecture classique (2/2)

Meilleur algo classique :

Cryptanalyse de ces standards sur une architecture classique (2/2)

Meilleur algo classique :

- RSA, DH $\longrightarrow$ GNFS, complexité $L[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}] = e^{\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln N)^{1/3}(\ln \ln N)^{2/3}}$
  - $\rightarrow$ par ex. RSA 2048 : $\ln N = 2048$, complexité du GNFS : $\approx 2^{112}$ opérations élementaires

Cryptanalyse de ces standards sur une architecture classique (2/2)

Meilleur algo classique :

- RSA, DH $\longrightarrow$ GNFS, complexité $L[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}] = e^{\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln N)^{1/3}(\ln \ln N)^{2/3}}$
  - $\rightarrow$ par ex. RSA 2048 : $\ln N = 2048$, complexité du GNFS : $\approx 2^{112}$ opérations élementaires
- ECDH $\longrightarrow$ Pollard ρ, complexité $\approx \mathcal{O}\left(\sqrt{\ln p}\right)$
  - $\rightarrow$ par ex. ECDSA 224 : $\ln p = 224$, complexité ρ-Pollard : $\approx 2^{112}$ opérations élementaires

Cryptanalyse de ces standards sur une architecture classique (2/2)

Meilleur algo classique :

- RSA, DH $\longrightarrow$ GNFS, complexité $L[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}] = e^{\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln N)^{1/3}(\ln\ln N)^{2/3}}$
  - $\rightarrow$ par ex. RSA 2048 : $\ln N = 2048$, complexité du GNFS : $\approx 2^{112}$ opérations élementaires
- ECDH $\longrightarrow$ Pollard ρ, complexité $\approx \mathcal{O}\left(\sqrt{\ln p}\right)$
  - $\rightarrow$ par ex. ECDSA 224 : $\ln p = 224$, complexité ρ-Pollard : $\approx 2^{112}$ opérations élementaires

Pour les primitives symétriques, les meilleures attaques ne font pas beaucoup mieux que du bruteforce :

Cryptanalyse de ces standards sur une architecture classique (2/2)

Meilleur algo classique :

- RSA, DH $\longrightarrow$ GNFS, complexité $L[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}] = e^{\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln N)^{1/3}(\ln \ln N)^{2/3}}$
  - $\rightarrow$ par ex. RSA 2048 : $\ln N = 2048$, complexité du GNFS : $\approx 2^{112}$ opérations élementaires
- ECDH $\longrightarrow$ Pollard ρ, complexité $\approx \mathcal{O}\left(\sqrt{\ln p}\right)$
  - $\rightarrow$ par ex. ECDSA 224 : $\ln p = 224$, complexité ρ-Pollard : $\approx 2^{112}$ opérations élementaires

Pour les primitives symétriques, les meilleures attaques ne font pas beaucoup mieux que du bruteforce :

- AES-128 sécurité d'environ 126 bits, plus de doutes sur AES-256

Cryptanalyse de ces standards sur une architecture classique (2/2)

Meilleur algo classique :

- RSA, DH $\longrightarrow$ GNFS, complexité $L[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}] = e^{\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln N)^{1/3}(\ln \ln N)^{2/3}}$
  - $\rightarrow$ par ex. RSA 2048 : $\ln N = 2048$, complexité du GNFS : $\approx 2^{112}$ opérations élementaires
- ECDH $\longrightarrow$ Pollard ρ, complexité $\approx \mathcal{O}\left(\sqrt{\ln p}\right)$
  - $\rightarrow$ par ex. ECDSA 224 : $\ln p = 224$, complexité ρ-Pollard : $\approx 2^{112}$ opérations élementaires

Pour les primitives symétriques, les meilleures attaques ne font pas beaucoup mieux que du bruteforce :

- AES-128 sécurité d'environ 126 bits, plus de doutes sur AES-256
- Chacha20 pas d'attaque significative mais algo plus récent, et moins étudié

Outline

1 Contexte

2 Motivation *aka.* la menace quantique
- Rappels de cryptographie
- L'ordinateur quantique et son impact sur la crypto actuelle

3 Cryptographie post-quantique (si on a le temps...)

4 Conclusion

Quantum computing

Quantum computing has been suggested by Paul Benioff in 1980. He proposed a quantum mechanical model of the Turing machine, using two ingredients:

Quantum computing

Quantum computing has been suggested by Paul Benioff in 1980. He proposed a quantum mechanical model of the Turing machine, using two ingredients:

- **Superposition**: while a bit can be either in a state 0 or 1, a quantum bit (*qubit*) can be in any *superposition* of states $|0\rangle$ and $|1\rangle$.

Quantum computing

Quantum computing has been suggested by Paul Benioff in 1980. He proposed a quantum mechanical model of the Turing machine, using two ingredients:

- **Superposition**: while a bit can be either in a state 0 or 1, a quantum bit (*qubit*) can be in any *superposition* of states $|0\rangle$ and $|1\rangle$.
- **Entanglement**: the capability of two qubits to be *correlated*. If Alice and Bob both get one of two entangled qubits, and if Alice measures a $|0\rangle$ at some point, then necessarily Bob must measure the same, as $|00\rangle$ is the only state where Alice's qubit is a $|0\rangle$.

Quantum computing

Quantum computing has been suggested by Paul Benioff in 1980. He proposed a quantum mechanical model of the Turing machine, using two ingredients:

- **Superposition**: while a bit can be either in a state 0 or 1, a quantum bit (*qubit*) can be in any *superposition* of states $|0\rangle$ and $|1\rangle$.
- **Entanglement**: the capability of two qubits to be *correlated*. If Alice and Bob both get one of two entangled qubits, and if Alice measures a $|0\rangle$ at some point, then necessarily Bob must measure the same, as $|00\rangle$ is the only state where Alice's qubit is a $|0\rangle$.

Qubits can be "implemented" using the spin of an electron, or the polarization of a photon, ...

Quantum computing

As a consequence:

- a vector of n entangled qubits can be in a superposition of any 2^n possible states **at the same time**,

Quantum computing

As a consequence:

- a vector of n entangled qubits can be in a superposition of any 2^n possible states **at the same time**,
- against 1 among the 2^n possible states for a classical vector of n bits.

Quantum computing

As a consequence:

- a vector of n entangled qubits can be in a superposition of any 2^n possible states **at the same time**,
- against 1 among the 2^n possible states for a classical vector of n bits.

It is however **not possible** to observe these states all together at the same time.

Quantum computing

As a consequence:

- a vector of n entangled qubits can be in a superposition of any 2^n possible states **at the same time**,
- against 1 among the 2^n possible states for a classical vector of n bits.

It is however **not possible** to observe these states all together at the same time.

A quantum algorithm solving a problem needs to make the correct solution (state) **exponentially** more likely than the other states (cf. quantum annealing / wave function collapsing).

Shor's algorithm [Sho97]

SIAM J. COMPUT.
Vol. 26, No. 5, pp. 1484–1509, October 1997

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.

Shor's algorithm: how it works

Algorithm 1: ShorAlgorithm(N)

Input: N

Output: p, q such that $N = pq$

Shor's algorithm: how it works

Algorithm 1: ShorAlgorithm(N)

Input: N

Output: p, q such that $N = pq$

Pick $g \in \mathbb{Z}_N$ at random;

Shor's algorithm: how it works

Algorithm 1: ShorAlgorithm(N)

Input: N

Output: p, q such that $N = pq$

```
Pick g ∈ Z_N at random;
if gcd(g, N) ≠ 1 then
    then return (p = gcd(g, N), q = N/p)
```

ENAC

Shor's algorithm: how it works

Algorithm 1: ShorAlgorithm(N)

Input: N

Output: p, q such that $N = pq$

Pick $g \in \mathbb{Z}_N$ at random;

if $\gcd(g, N) \neq 1$ **then**

  then **return** $(p = \gcd(g, N), q = N/p)$

Find r such that $g^r \equiv 1[N]$;

Shor's algorithm: how it works

Algorithm 1: ShorAlgorithm(N)

Input: N

Output: p, q such that $N = pq$

1 Pick $g \in \mathbb{Z}_N$ at random;

2 **if** $\gcd(g, N) \neq 1$ **then**

3 then **return** $(p = \gcd(g, N), q = N/p)$

4 Find r such that $g^r \equiv 1[N]$;

5 **if** $r \equiv 0[2]$ **then**

6 **return** $\gcd\left(g^{r/2} \pm 1, N\right)$

7 **else**

8 go to 1

Shor's algorithm: how it works

"Find r such that $g^r \equiv 1[N]$;"

Shor's algorithm: how it works

"Find r such that $g^r \equiv 1[N]$;"

- First question: How does finding r such that $g^r \equiv 1[N]$ help factoring?

Shor's algorithm: how it works

"Find r such that $g^r \equiv 1[N]$;"

- First question: How does finding r such that $g^r \equiv 1[N]$ help factoring?

$$g^r \equiv 1[N] \quad \Leftrightarrow \quad \exists k \in \mathbb{N}^* \text{ such that } g^r = kN + 1$$

Shor's algorithm: how it works

"Find r such that $g^r \equiv 1[N]$;"

- First question: How does finding r such that $g^r \equiv 1[N]$ help factoring?

$$g^r \equiv 1[N] \quad \Leftrightarrow \quad \exists k \in \mathbb{N}^* \text{ such that } g^r = kN + 1$$
$$\Leftrightarrow \quad g^r - 1 = kN$$

Shor's algorithm: how it works

"Find r such that $g^r \equiv 1[N]$;"

- First question: How does finding r such that $g^r \equiv 1[N]$ help factoring?

$$\begin{aligned} g^r \equiv 1[N] &\Leftrightarrow \exists k \in \mathbb{N}^* \text{ such that } g^r = kN + 1 \\ &\Leftrightarrow g^r - 1 = kN \\ \textbf{(assuming } r \textbf{ is even)} &\Leftrightarrow \left(g^{r/2} - 1\right)\left(g^{r/2} + 1\right) = kN \end{aligned}$$

Shor's algorithm: how it works

"Find r such that $g^r \equiv 1[N]$;"

- First question: How does finding r such that $g^r \equiv 1[N]$ help factoring?

$$\begin{aligned} g^r \equiv 1[N] \quad &\Leftrightarrow \quad \exists k \in \mathbb{N}^* \text{ such that } g^r = kN + 1 \\ &\Leftrightarrow \quad g^r - 1 = kN \\ \textbf{(assuming } r \textbf{ is even)} \quad &\Leftrightarrow \quad \left(g^{r/2} - 1\right)\left(g^{r/2} + 1\right) = kN \end{aligned}$$

Meaning that there is a non-negligible probability that $g^{r/2} \pm 1$ shares non trivial factors with N.

Shor's algorithm: how it works

Example with $N = 314191$, find p, q

(source: minutephysics)

Shor's algorithm: how it works

Example with $N = 314191$, find p, q

(source: minutephysics)

- step 1. $g \leftarrow 101$

Shor's algorithm: how it works

Example with $N = 314191$, find p, q

(source: minutephysics)

- step 1. $g \leftarrow 101$
- step 2. $r \leftarrow 4347$

Shor's algorithm: how it works

Example with $N = 314191$, find p, q

(source: minutephysics)

- step 1. $g \leftarrow 101$
- step 2. $r \leftarrow 4347$
- step 3. r is **odd**... go to 1

Shor’s algorithm: how it works

Example with $N = 314191$, find p, q **(source: minutephysics)**

- step 1. $g \leftarrow 101$
- step 2. $r \leftarrow 4347$
- step 3. r is **odd**... go to 1
- step 1. $g \leftarrow 127$

Shor's algorithm: how it works

Example with $N = 314191$, find p, q **(source: minutephysics)**

- step 1. $g \leftarrow 101$
- step 2. $r \leftarrow 4347$
- step 3. r is **odd**... go to 1
- step 1. $g \leftarrow 127$
- step 2. $r \leftarrow 17388$

Shor's algorithm: how it works

Example with $N = 314191$, find p, q

(source: minutephysics)

- step 1. $g \leftarrow 101$
- step 2. $r \leftarrow 4347$
- step 3. r is **odd**... go to 1
- step 1. $g \leftarrow 127$
- step 2. $r \leftarrow 17388$
- step 3. let us denote $g_p = g^{17388/2} + 1$ and $g_q = g^{17388/2} - 1$

Shor's algorithm: how it works

Example with $N = 314191$, find p, q

(source: minutephysics)

- step 1. $g \leftarrow 101$
- step 2. $r \leftarrow 4347$
- step 3. r is **odd**... go to 1
- step 1. $g \leftarrow 127$
- step 2. $r \leftarrow 17388$
- step 3. let us denote $g_p = g^{17388/2} + 1$ and $g_q = g^{17388/2} - 1$
  we have that $\gcd(g_p, N) = 829 =: p$ and $\gcd(g_q, N) = 379 =: q$
  and indeed, $p \cdot q = 829 \times 379 = 314191 = N$

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

"Find r such that $g^r \equiv 1[N]$;"

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

"**Quantumly** find r such that $g^r \equiv 1[N]$;"

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

"**Quantumly** find r such that $g^r \equiv 1[N]$;"

The complexity to find the *period* of the function $g \mapsto g^x \mod N$ is:

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

"**Quantumly** find r such that $g^r \equiv 1[N]$;"

The complexity to find the *period* of the function $g \mapsto g^x \mod N$ is:

- Classically $\mathcal{O}(N)$

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

"**Quantumly** find r such that $g^r \equiv 1[N]$;"

The complexity to find the *period* of the function $g \mapsto g^x \mod N$ is:

- Classically $\mathcal{O}(N)$ (which is exponential in the size of the input $\log(N)$)

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

"**Quantumly** find r such that $g^r \equiv 1[N]$;"

The complexity to find the *period* of the function $g \mapsto g^x \mod N$ is:

- Classically $\mathcal{O}(N)$ (which is exponential in the size of the input $\log(N)$)
- Quantumly $\mathcal{O}\left(\log(N)^3\right)$

Shor's algorithm: how it works

- Second question: "*Wait a minute, I was expecting some magic quantum trick out there. Where the is the quantum part?*"

"**Quantumly** find r such that $g^r \equiv 1[N]$;"

The complexity to find the *period* of the function $g \mapsto g^x \mod N$ is:

- Classically $\mathcal{O}(N)$ (which is exponential in the size of the input $\log(N)$)
- Quantumly $\mathcal{O}\left(\log(N)^3\right)$ (polynomial in the size of the input): That's an **exponential** speedup!

Quantum period finding

How does it work? Why is it much much faster quantumly?

Fourier Transform is THE tool to analyse frequencies. Fortunately, it has a quantum equivalent: QFT.

Quantum computing allows to provide QFT a superposition of every possible states (assuming enough qubits).

Consequences of Shor's algorithm on PKC

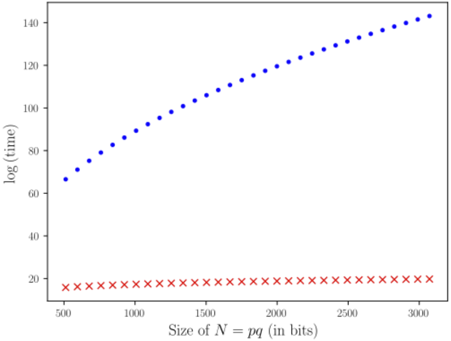
- Factoring becomes polynomial-time
  $\mathcal{O}\left((\log N)^2 (\log \log N) (\log \log \log N)\right)$

Consequences of Shor's algorithm on PKC

- Factoring becomes polynomial-time
  $\mathcal{O}\left((\log N)^2 (\log\log N)(\log\log\log N)\right)$
  against $\exp\left(1.9(\log N)^{1/3}(\log\log N)^{2/3}\right)$
  classically

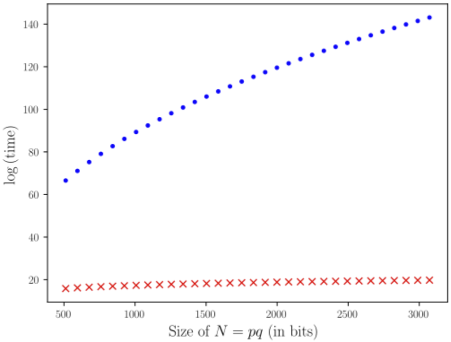
Consequences of Shor's algorithm on PKC

- Factoring becomes polynomial-time $\mathcal{O}\left((\log N)^2 (\log\log N)(\log\log\log N)\right)$ against $\exp\left(1.9(\log N)^{1/3}(\log\log N)^{2/3}\right)$ classically



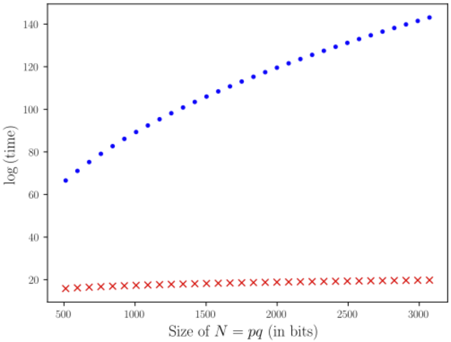
Consequences of Shor's algorithm on PKC

- Factoring becomes polynomial-time $\mathcal{O}\left((\log N)^2 (\log\log N)(\log\log\log N)\right)$ against $\exp\left(1.9(\log N)^{1/3}(\log\log N)^{2/3}\right)$ classically
- Discrete logarithm becomes almost polynomial-time



Consequences of Shor's algorithm on PKC

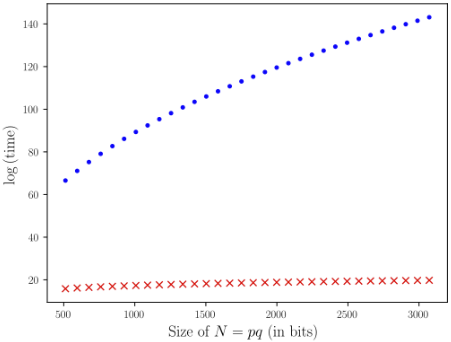
- Factoring becomes polynomial-time $\mathcal{O}\left((\log N)^2 (\log\log N)(\log\log\log N)\right)$ against $\exp\left(1.9(\log N)^{1/3}(\log\log N)^{2/3}\right)$ classically
- Discrete logarithm becomes almost polynomial-time



No more RSA, DSA, ECDSA, ElGamal, ...

Consequences of Shor's algorithm on PKC

- Factoring becomes polynomial-time $\mathcal{O}\left((\log N)^2 (\log\log N)(\log\log\log N)\right)$ against $\exp\left(1.9(\log N)^{1/3}(\log\log N)^{2/3}\right)$ classically
- Discrete logarithm becomes almost polynomial-time



No more RSA, DSA, ECDSA, ElGamal, ...

In other words, **security as we know it collapses**...

Outline

Clarification

What are the alternatives to classical cryptography in presence of an adversary equipped with a large scale quantum computer?

Clarification

What are the alternatives to classical cryptography in presence of an adversary equipped with a large scale quantum computer?

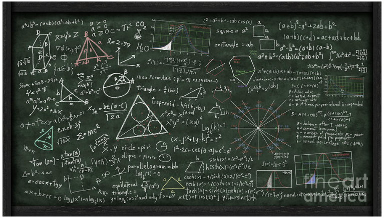
- Quantum Key Exchange (out of the scope of this talk)

Clarification

What are the alternatives to classical cryptography in presence of an adversary equipped with a large scale quantum computer?

- Quantum Key Exchange (out of the scope of this talk)

- Post-Quantum Cryptography



L'essentiel

Il existe 5 primitives populaires pour construire des problèmes *a priori* difficiles (et donc exploitables en crypto) y compris pour un ordinateur quantique :

- les réseaux euclidiens (EN: lattices)
- les codes correcteurs d'erreurs
- les fonctions de hachage
- les polynômes multivariés
- les isogénies entre courbes elliptiques super-singulières

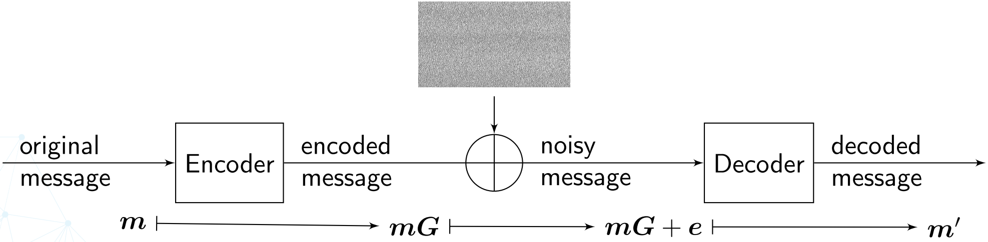
L'essentiel

Il existe 5 primitives populaires pour construire des problèmes *a priori* difficiles (et donc exploitables en crypto) y compris pour un ordinateur quantique :

- les réseaux euclidiens (EN: lattices)
- **les codes correcteurs d'erreurs**
- les fonctions de hachage
- les polynômes multivariés
- les isogénies entre courbes elliptiques super-singulières

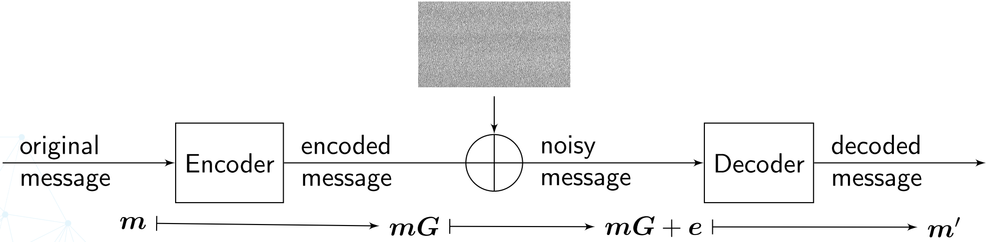
Coding theory

Coding theory is the science of (efficiently) adding redundancy to information in order to detect/correct errors that could occur during transmission.



Coding theory

Coding theory is the science of (efficiently) adding redundancy to information in order to detect/correct errors that could occur during transmission.

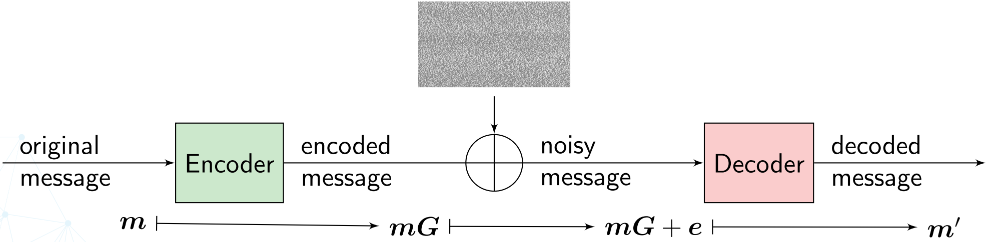


Preliminary remarks:

- Hopefully, we have $m' = m$

Coding theory

Coding theory is the science of (efficiently) adding redundancy to information in order to detect/correct errors that could occur during transmission.



Preliminary remarks:

- Hopefully, we have $m' = m$
- For code-based PKC, most of the time, public encoder / private decoder.

Definitions

Linear code

A *linear code* of dimension k and length n over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$.

A linear code $\mathcal{C}[n, k]$ is fully determined by one of the following matrices:

Definitions

Linear code

A *linear code* of dimension k and length n over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$.

A linear code $\mathcal{C}[n,k]$ is fully determined by one of the following matrices:

Generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$

$\mathcal{C} = \left\{ \mathbf{xG}, \text{ for } \mathbf{x} \in \mathbb{F}_q^k \right\}$

Definitions

Linear code

A *linear code* of dimension k and length n over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$.

A linear code $\mathcal{C}[n, k]$ is fully determined by one of the following matrices:

Generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$	Parity-check matrix $\boldsymbol{H} \in \mathbb{F}_q^{(n-k) \times n}$
$\mathcal{C} = \left\{\mathbf{xG}, \text{ for } \mathbf{x} \in \mathbb{F}_q^k\right\}$	$\mathcal{C} = \left\{\boldsymbol{s} \in \mathbb{F}_q^n \text{ such that } \boldsymbol{H}\boldsymbol{s}^\top = \mathbf{0}\right\}$

Definitions

Linear code

A *linear code* of dimension k and length n over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$.

A linear code $\mathcal{C}[n,k]$ is fully determined by one of the following matrices:

Generator matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$	Parity-check matrix $\boldsymbol{H} \in \mathbb{F}_q^{(n-k) \times n}$
$\mathcal{C} = \left\{ \mathbf{xG}, \text{ for } \mathbf{x} \in \mathbb{F}_q^k \right\}$	$\mathcal{C} = \left\{ \boldsymbol{s} \in \mathbb{F}_q^n \text{ such that } \boldsymbol{H}\boldsymbol{s}^\top = \mathbf{0} \right\}$

The Hamming weight of a word $\boldsymbol{u}$ is the number of its non-zero coordinates:

$$wt\left(\boldsymbol{u}\right) = \#\left\{ i \in \{0, \ldots, n-1\} \text{ such that } \boldsymbol{u}_i \neq 0 \right\}$$

$$\text{example : } wt\left((0,1,0,0,1,0,1,0)\right) = 3$$

Code-based cryptography (CBC)

Que sont les codes correcteurs ?

Code-based cryptography (CBC)

Que sont les codes correcteurs ?

Des façons de rajouter de la redondance à l'information utile, afin d'être capable de détecter — voire corriger — d'éventuelles erreurs lors de la transmission.

Code-based cryptography (CBC)

Que sont les codes correcteurs ?

Des façons de rajouter de la redondance à l'information utile, afin d'être capable de détecter — voire corriger — d'éventuelles erreurs lors de la transmission.

Exemple : le code à répétition

Message à envoyer		1			0			1	
Encodage	1	1	1	0	0	0	1	1	1
Message reçu	0	1	1	0	1	0	1	1	0
Message décodé		1			0			1	

Code-based cryptography (CBC)

Que sont les codes correcteurs ?

Des façons de rajouter de la redondance à l'information utile, afin d'être capable de détecter — voire corriger — d'éventuelles erreurs lors de la transmission.

Exemple : le code à répétition

Message à envoyer		1			0			1	
Encodage	1	1	1	0	0	0	1	1	1
Message reçu	0	1	1	0	1	0	1	1	0
Message décodé		1			0			1	

Ce code est particulièrement mauvais (bien qu'utile pédagogiquement parlant) :

- dimension : $k = 1$
- longueur : $n = 3$
- distance minimale : $d = 3$

- capacité de détection : $d - 1 = 2$ erreurs
- capacité de correction : $\left\lfloor \frac{d-1}{2} \right\rfloor = 1$ erreur
- rendement $\frac{k}{n} = \frac{1}{3}$.

Code-based cryptography (CBC)

Problème du décodage de syndrome.

Code-based cryptography (CBC)

Problème du décodage de syndrome.

Problème

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$.

Code-based cryptography (CBC)

Problème du décodage de syndrome.

Problème

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$.

Ce problème est-il difficile ?

Code-based cryptography (CBC)

Problème du décodage de syndrome.

Problème

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$.

Ce problème est-il difficile ? **non !**

Code-based cryptography (CBC)

Problème du décodage de syndrome.

Problème

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$.

Ce problème est-il difficile ? non !
Il suffit de réaliser un pivot de Gauss sur la matrice $\mathbf{H}$. C'est purement un problème d'algèbre linéaire...

Code-based cryptography (CBC)

Problème du décodage de syndrome.

Problème

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$.

Ce problème est-il difficile ? non !
Il suffit de réaliser un pivot de Gauss sur la matrice $\mathbf{H}$. C'est purement un problème d'algèbre linéaire...

Problème modifié

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$**, et $\mathbf{x}$ de poids relativement faible**.

Code-based cryptography (CBC)

Problème du décodage de syndrome.

Problème

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$.

Ce problème est-il difficile ? non !
Il suffit de réaliser un pivot de Gauss sur la matrice $\mathbf{H}$. C'est purement un problème d'algèbre linéaire...

Problème modifié

Soit $\mathbf{s} \in \mathbb{F}_2^{n-k}$ et $\mathbf{H} \in \mathbb{F}_2^{(n-k)\times n}$. Trouver $\mathbf{x} \in \mathbb{F}_2^n$ tel que $\mathbf{H}\mathbf{x}^\top = \mathbf{s}$, **et $\mathbf{x}$ de poids relativement faible**.

Le problème devient *NP*-difficile [BMvT78].

(Traduction: il devient cryptographiquement intéressant)

Code-based cryptography (CBC)

Cryptosystème de McEliece [McE78]

Code-based cryptography (CBC)

Cryptosystème de McEliece [McE78]
Soit $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ la matrice génératrice d'un code (de Goppa binaire) $\mathcal{C}$ pouvant corriger jusqu'à t erreurs à l'aide de l'algorithme de décodage $\mathcal{D}_{\mathbf{G}}$.

Code-based cryptography (CBC)

Cryptosystème de McEliece [McE78]
Soit $\mathbf{G} \in \mathbb{F}_2^{k \times n}$ la matrice génératrice d'un code (de Goppa binaire) $\mathcal{C}$ pouvant corriger jusqu'à t erreurs à l'aide de l'algorithme de décodage $\mathcal{D}_{\mathbf{G}}$.

matrice inversible $\mathbf{S} \in \mathbb{F}_2^{k \times k}$

message $\mathbf{m} \in \mathbb{F}_2^{k}$

matrice permutation $\mathbf{P} \in \mathbb{F}_2^{n \times n}$

$$\xrightarrow{\tilde{\mathbf{G}}=\mathbf{SGP}, n, k, t}$$

$\mathbf{e} \in \mathbb{F}_2^{n}$ tel que $wt(\mathbf{e}) \leq t$

$$\xleftarrow{\mathbf{c}}$$

$$\mathbf{c} = \mathbf{m}\tilde{\mathbf{G}} + \mathbf{e}$$

$$\tilde{\mathbf{c}} = \mathcal{D}_{\mathbf{G}}\left(\mathbf{c}\mathbf{P}^{-1}\right) = \mathcal{D}_{\mathbf{G}}\left(\mathbf{mSG} + \mathbf{eP}^{-1}\right)$$

$$\mathbf{m} = \tilde{\mathbf{c}}\mathbf{S}^{-1}$$

CBC : un exemple

Soit $\mathcal{C}$ le code (de Hamming) admettant pour matrice de parité $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Soit $\mathbf{v} = (1, 0, 0, 0, 1, 1, 1)$ le mot reçu. Quel était le message envoyé ?

CBC : un exemple

Soit $\mathcal{C}$ le code (de Hamming) admettant pour matrice de parité $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Soit $\mathbf{v} = (1, 0, 0, 0, 1, 1, 1)$ le mot reçu. Quel était le message envoyé ? Décodons

CBC : un exemple

Soit $\mathcal{C}$ le code (de Hamming) admettant pour matrice de parité $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Soit $\mathbf{v} = (1, 0, 0, 0, 1, 1, 1)$ le mot reçu. Quel était le message envoyé ? Décodons

$$\mathbf{v} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

CBC : un exemple

Soit $\mathcal{C}$ le code (de Hamming) admettant pour matrice de parité $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Soit $\mathbf{v} = (1, 0, 0, 0, 1, 1, 1)$ le mot reçu. Quel était le message envoyé ? Décodons

$$\mathbf{v} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \mathbf{s}$$

CBC : un exemple

Soit $\mathcal{C}$ le code (de Hamming) admettant pour matrice de parité $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Soit $\mathbf{v} = (1, 0, 0, 0, 1, 1, 1)$ le mot reçu. Quel était le message envoyé ? Décodons

$$\mathbf{v} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & \color{red}{1} & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & \color{red}{1} & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} \color{red}{1} \\ 0 \\ \color{red}{1} \end{pmatrix} = \mathbf{s}$$

CBC : un exemple

Soit $\mathcal{C}$ le code (de Hamming) admettant pour matrice de parité $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Soit $\mathbf{v} = (1, 0, 0, 0, 1, 1, 1)$ le mot reçu. Quel était le message envoyé ? Décodons

$$\mathbf{v} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \mathbf{e} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \mathbf{s}$$

CBC : un exemple

Soit $\mathcal{C}$ le code (de Hamming) admettant pour matrice de parité $\mathbf{H}$:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Soit $\mathbf{v} = (1,0,0,0,1,1,1)$ le mot reçu. Quel était le message envoyé ? Décodons

$$\mathbf{v} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \mathbf{e} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = \mathbf{s}$$

$$\mathbf{m} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

www.enac.fr

Quasi-Cyclic Moderate Density Parity-Check Codes

KeyGen

Sample $\mathbf{h}_0, \mathbf{h}_1 \leftarrow \mathbb{F}_2^r$ of small weight w, $\mathbf{h}_0$ inversible. Compute $\mathbf{h} = \mathbf{h}_1 \mathbf{h}_0^{-1}$.

$$\mathbf{H}_{\text{secret}} = \left(\begin{array}{c|c} \mathbf{h}_0 & \mathbf{h}_1 \\ \circlearrowright & \circlearrowright \end{array} \right)$$

$$\mathbf{H}_{\text{pub}} = \left(\begin{array}{c|c} (1, 0, \ldots, 0) & \mathbf{h} \\ \circlearrowright & \circlearrowright \end{array} \right)$$

Quasi-Cyclic Moderate Density Parity-Check Codes

KeyGen

Sample $\mathbf{h}_0, \mathbf{h}_1 \leftarrow \mathbb{F}_2^r$ of small weight w, $\mathbf{h}_0$ inversible. Compute $\mathbf{h} = \mathbf{h}_1 \mathbf{h}_0^{-1}$.

$$\mathbf{H}_{\text{secret}} = \left(\begin{array}{c|c} \underset{\circlearrowright}{\mathbf{h}_0} & \underset{\circlearrowright}{\mathbf{h}_1} \end{array} \right)$$

$$\mathbf{H}_{\text{pub}} = \left(\begin{array}{c|c} \underset{\circlearrowright}{(1, 0, \ldots, 0)} & \underset{\circlearrowright}{\mathbf{h}} \end{array} \right)$$

Encryption

As for McEliece, $\mathbf{e}$ of weight t,

$$\mathbf{c} = \mathbf{mG} + \mathbf{e}.$$

Quasi-Cyclic Moderate Density Parity-Check Codes

KeyGen

Sample $\mathbf{h}_0, \mathbf{h}_1 \leftarrow \mathbb{F}_2^r$ of small weight w, $\mathbf{h}_0$ inversible. Compute $\mathbf{h} = \mathbf{h}_1 \mathbf{h}_0^{-1}$.

$$\mathbf{H}_{\text{secret}} = \left(\begin{array}{c|c} \mathbf{h}_0 & \mathbf{h}_1 \\ \circlearrowleft & \circlearrowleft \end{array} \right)$$

$$\mathbf{H}_{\text{pub}} = \left(\begin{array}{c|c} (1, 0, \dots, 0) & \mathbf{h} \\ \circlearrowleft & \circlearrowleft \end{array} \right)$$

Encryption

As for McEliece, $\mathbf{e}$ of weight t,

$$\mathbf{c} = \mathbf{m}\mathbf{G} + \mathbf{e}.$$

Decryption

Use an iterative decoder (e.g. the BitFlipping algorithm) to recover message $\mathbf{m}$.

Quasi-Cyclic Moderate Density Parity-Check Codes

KeyGen

Sample $\mathbf{h}_0, \mathbf{h}_1 \leftarrow \mathbb{F}_2^r$ of small weight w, $\mathbf{h}_0$ inversible. Compute $\mathbf{h} = \mathbf{h}_1 \mathbf{h}_0^{-1}$.

$$\mathbf{H}_{\text{secret}} = \left(\begin{array}{c|c} \mathbf{h}_0 & \mathbf{h}_1 \\ \circlearrowright & \circlearrowright \end{array} \right)$$

$$\mathbf{H}_{\text{pub}} = \left(\begin{array}{c|c} (1, 0, \ldots, 0) & \mathbf{h} \\ \circlearrowright & \circlearrowright \end{array} \right)$$

Encryption

As for McEliece, $\mathbf{e}$ of weight t,

$$\mathbf{c} = \mathbf{m}\mathbf{G} + \mathbf{e}.$$

Decryption

Use an iterative decoder (e.g. the BitFlipping algorithm) to recover message $\mathbf{m}$.

Suggested parameters: $r = 9857, n = 2r, w = 142, t = 134$ for 128 bits.

Quasi-Cyclic Moderate Density Parity-Check Codes

KeyGen

Sample $\mathbf{h}_0, \mathbf{h}_1 \leftarrow \mathbb{F}_2^r$ of small weight w, $\mathbf{h}_0$ inversible. Compute $\mathbf{h} = \mathbf{h}_1 \mathbf{h}_0^{-1}$.

$$\mathbf{H}_{\text{secret}} = \left(\begin{array}{c|c} \mathbf{h}_0 & \mathbf{h}_1 \\ \circlearrowright & \circlearrowright \end{array} \right)$$

$$\mathbf{H}_{\text{pub}} = \left(\begin{array}{c|c} (1, 0, \ldots, 0) & \mathbf{h} \\ \circlearrowright & \circlearrowright \end{array} \right)$$

Encryption

As for McEliece, $\mathbf{e}$ of weight t,

$$\mathbf{c} = \mathbf{m}\mathbf{G} + \mathbf{e}.$$

Decryption

Use an iterative decoder (e.g. the BitFlipping algorithm) to recover message $\mathbf{m}$.

Suggested parameters: $r = 9857, n = 2r, w = 142, t = 134$ for 128 bits. Resulting sizes?

Code-based cryptography (CBC)

Avantage

Code-based cryptography (CBC)

Avantage

- Très efficace (algèbre linéaire)

Code-based cryptography (CBC)

Avantage

- Très efficace (algèbre linéaire)
- Arithmétique simple (modulo 2 vs modulo $2^{\sim 1024}$ pour RSA)

Code-based cryptography (CBC)

Avantage

- Très efficace (algèbre linéaire)
- Arithmétique simple (modulo 2 vs modulo $2^{\sim 1024}$ pour RSA)
- Hautement parallélisable

Code-based cryptography (CBC)

Avantage

- Très efficace (algèbre linéaire)
- Arithmétique simple (modulo 2 vs modulo $2^{\sim 1024}$ pour RSA)
- Hautement parallélisable

Inconvénients

- Taille de clés conséquente... (quasi-cyclique ?)

Code-based cryptography (CBC)

Avantage

- Très efficace (algèbre linéaire)
- Arithmétique simple (modulo 2 vs modulo $2^{\sim 1024}$ pour RSA)
- Hautement parallélisable

Inconvénients

- Taille de clés conséquente... (quasi-cyclique ?)
- Hypothèse d'indistingabilité de la famille de codes utilisée (plus technique)

Code-based cryptography (CBC)

Avantage

- Très efficace (algèbre linéaire)
- Arithmétique simple (modulo 2 vs modulo $2^{\sim 1024}$ pour RSA)
- Hautement parallélisable

Inconvénients

- Taille de clés conséquente... (quasi-cyclique ?)
- Hypothèse d'indistingabilité de la famille de codes utilisée (plus technique)

Chiffrement OK. Existe-t-il un algo de signature aussi simple ?

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w

Clé publique $\mathbf{H}$ et le syndrome de la clé secrète $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w
Clé publique $\mathbf{H}$ et le syndrome de la clé secrète $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$

message m

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w

Clé publique $\mathbf{H}$ et le syndrome de la clé secrète $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$

message m

$\mathbf{y}$ de poids faible

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w

Clé publique $\mathbf{H}$ et le syndrome de la clé secrète $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$

message m

$\mathbf{y}$ de poids faible

$\mathbf{c} = \mathcal{H}\left(\mathbf{H}\mathbf{y}^\top, \mathbf{m}\right)$ de poids faible

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w
Clé publique $\mathbf{H}$ et le syndrome de la clé secrète $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$

message m

$\mathbf{y}$ de poids faible

$\mathbf{c} = \mathcal{H}\left(\mathbf{H}\mathbf{y}^\top, \mathbf{m}\right)$ de poids faible

$$\mathbf{z} = \mathbf{x} \cdot \mathbf{c} + \mathbf{y}$$

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w
Clé publique $\mathbf{H}$ et le syndrome de la clé secrète $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$

message m
$\mathbf{y}$ de poids faible
$\mathbf{c} = \mathcal{H}\left(\mathbf{H}\mathbf{y}^\top, \mathbf{m}\right)$ de poids faible

$$\mathbf{z} = \mathbf{x} \cdot \mathbf{c} + \mathbf{y} \quad \xrightarrow{\mathbf{z},\mathbf{c}}$$

Code-based cryptography (CBC) : Exemple de signature efficace

Clé secrète $\mathbf{x}$ de poids faible w

Clé publique $\mathbf{H}$ et le syndrome de la clé secrète $\mathbf{s} = \mathbf{H}\mathbf{x}^\top$

message m

$\mathbf{y}$ de poids faible

$\mathbf{c} = \mathcal{H}\left(\mathbf{H}\mathbf{y}^\top, \mathbf{m}\right)$ de poids faible

$\mathbf{z} = \mathbf{x} \cdot \mathbf{c} + \mathbf{y}$ $\xrightarrow{\mathbf{z},\mathbf{c}}$ Verif ?

Code-based cryptography (CBC) : Exemple de signature efficace

Verif :

Code-based cryptography (CBC) : Exemple de signature efficace

Verif :

- $wt(\mathbf{z}) \leq \tilde{w}$ pas trop grand

Code-based cryptography (CBC) : Exemple de signature efficace

Verif :

- $wt\left(\mathbf{z}\right) \leq \tilde{w}$ pas trop grand
- Vérifier que $\mathcal{H}\left(\mathbf{H}\mathbf{z}^{\top} - \mathbf{s}\cdot\mathbf{c}, \mathbf{m}\right) == \mathbf{c}$

Code-based cryptography (CBC) : Exemple de signature efficace

Verif :

- $wt(\mathbf{z}) \leq \tilde{w}$ pas trop grand
- Vérifier que $\mathcal{H}\left(\mathbf{H}\mathbf{z}^\top - \mathbf{s} \cdot \mathbf{c}, \mathbf{m}\right) == \mathbf{c}$

En théorie, ça fonctionne. Mais en pratique...

Code-based cryptography (CBC) : Exemple de signature efficace

Verif :

- $wt(\mathbf{z}) \leq \tilde{w}$ pas trop grand
- Vérifier que $\mathcal{H}\left(\mathbf{H}\mathbf{z}^{\top} - \mathbf{s} \cdot \mathbf{c}, \mathbf{m}\right) == \mathbf{c}$

En théorie, ça fonctionne. Mais en pratique...

Le problème peut s'écrire sous forme d'un décodage de syndrome :

Code-based cryptography (CBC) : Exemple de signature efficace

Verif :

- $wt\left(\mathbf{z}\right) \leq \tilde{w}$ pas trop grand
- Vérifier que $\mathcal{H}\left(\mathbf{Hz}^{\top} - \mathbf{s} \cdot \mathbf{c}, \mathbf{m}\right) == \mathbf{c}$

En théorie, ça fonctionne. Mais en pratique...

Le problème peut s'écrire sous forme d'un décodage de syndrome :

$$\mathbf{z} = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & c_0 & c_1 & \dots & c_{n-1} \\ 0 & 1 & \dots & 0 & c_{n-1} & c_0 & \dots & c_{n-2} \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & 1 & c_1 & c_2 & \dots & c_0 \end{array}\right) \cdot \begin{pmatrix} \mathbf{y} \\ \mathbf{x} \end{pmatrix}$$

Code-based cryptography (CBC) : Exemple de signature efficace

Syndrome connu, matrice de parité creuse connue (LDPC)

Code-based cryptography (CBC) : Exemple de signature efficace

Syndrome connu, matrice de parité creuse connue (LDPC)
$\Rightarrow$ décodage classique facile et efficace

Code-based cryptography (CBC) : Exemple de signature efficace

Syndrome connu, matrice de parité creuse connue (LDPC)
$\Rightarrow$ décodage classique facile et efficace
$\Rightarrow$ cryptanalyse

Code-based cryptography (CBC) : Exemple de signature efficace

Syndrome connu, matrice de parité creuse connue (LDPC)
$\Rightarrow$ décodage classique facile et efficace
$\Rightarrow$ cryptanalyse

Claimed security	Persichetti's OTS parameters				xBF parameters		Verification	Cryptanalysis
	n	w_1	w_2	δ	τ	N	t_{verify} (ms)	t_{break} (ms)
80	4801	90	100	10	7	5	22.569	165.459
	3072	85	85	7	5	5	14.271	68.858
128	9857	150	200	12	9	10	99.492	453.680
	6272	125	125	10	7	10	42.957	288.442

Code-based cryptography (CBC) : Exemple de signature efficace

Syndrome connu, matrice de parité creuse connue (LDPC)
⇒ décodage classique facile et efficace
⇒ cryptanalyse

Claimed security	Persichetti's OTS parameters				xBF parameters		Verification	Cryptanalysis
	n	w_1	w_2	δ	τ	N	t_{verify} (ms)	t_{break} (ms)
80	4801	90	100	10	7	5	22.569	165.459
	3072	85	85	7	5	5	14.271	68.858
128	9857	150	200	12	9	10	99.492	453.680
	6272	125	125	10	7	10	42.957	288.442

D'autres schémas de signature (plus complexes à exposer) existent, et ne souffrent pas de ce type de problème:

- WAVE [DST18]: `https://eprint.iacr.org/2018/996`
- DURANDAL [ABG+18]: `https://eprint.iacr.org/2018/1192` **(métrique rang)**

Outline

Course conclusion

- Cryptography has reached a stable phase, where:

Course conclusion

- Cryptography has reached a stable phase, where:
  - Symmetric primitives are fast and secure (sufficiently attacked to be considered as such)

Course conclusion

- Cryptography has reached a stable phase, where:
  - Symmetric primitives are fast and secure (sufficiently attacked to be considered as such)
  - Asymmetric primitives have been sufficiently improved to be largely deployed

Course conclusion

- Cryptography has reached a stable phase, where:
  - Symmetric primitives are fast and secure (sufficiently attacked to be considered as such)
  - Asymmetric primitives have been sufficiently improved to be largely deployed
  - Hybrid encryption allows to benefit from SE efficiency while avoiding its disadvantages

Course conclusion

- Cryptography has reached a stable phase, where:
  - Symmetric primitives are fast and secure (sufficiently attacked to be considered as such)
  - Asymmetric primitives have been sufficiently improved to be largely deployed
  - Hybrid encryption allows to benefit from SE efficiency while avoiding its disadvantages
- There is a real quantum threat for actual cryptography.

Course conclusion

- Cryptography has reached a stable phase, where:
  - Symmetric primitives are fast and secure (sufficiently attacked to be considered as such)
  - Asymmetric primitives have been sufficiently improved to be largely deployed
  - Hybrid encryption allows to benefit from SE efficiency while avoiding its disadvantages
- There is a real quantum threat for actual cryptography.
- Post-quantum alternatives exist and are being developed/standardized (involve classical computers, not quantum).

Course conclusion

- Cryptography has reached a stable phase, where:
  - Symmetric primitives are fast and secure (sufficiently attacked to be considered as such)
  - Asymmetric primitives have been sufficiently improved to be largely deployed
  - Hybrid encryption allows to benefit from SE efficiency while avoiding its disadvantages
- There is a real quantum threat for actual cryptography.
- Post-quantum alternatives exist and are being developed/standardized (involve classical computers, not quantum).
- Most serious candidates are lattices, error correcting codes and hash functions.

Course conclusion

- Cryptography has reached a stable phase, where:
  - Symmetric primitives are fast and secure (sufficiently attacked to be considered as such)
  - Asymmetric primitives have been sufficiently improved to be largely deployed
  - Hybrid encryption allows to benefit from SE efficiency while avoiding its disadvantages
- There is a real quantum threat for actual cryptography.
- Post-quantum alternatives exist and are being developed/standardized (involve classical computers, not quantum).
- Most serious candidates are lattices, error correcting codes and hash functions.
- Keys for symmetric algorithms need to be doubled.

Upgoing developments for PQC

- NIST PQC standards should be ready by 2024.

Upgoing developments for PQC

- NIST PQC standards should be ready by 2024.
- Integration and deployment will probably take another 5-10 years

Upgoing developments for PQC

- NIST PQC standards should be ready by 2024.
- Integration and deployment will probably take another 5-10 years
- Some candidates have already been chosen:

Upgoing developments for PQC

- NIST PQC standards should be ready by 2024.
- Integration and deployment will probably take another 5-10 years
- Some candidates have already been chosen:
  - 1 lattice based KEM
  - 2 lattice based signatures
  - 1 hash based signature

Upgoing developments for PQC

- NIST PQC standards should be ready by 2024.
- Integration and deployment will probably take another 5-10 years
- Some candidates have already been chosen:
  - 1 lattice based KEM
  - 2 lattice based signatures
  - 1 hash based signature
- Fourth (last) round only features code based candidates

Upgoing developments for PQC

- NIST PQC standards should be ready by 2024.
- Integration and deployment will probably take another 5-10 years
- Some candidates have already been chosen:
  - 1 lattice based KEM
  - 2 lattice based signatures
  - 1 hash based signature
- Fourth (last) round only features code based candidates
- A call for additional signatures has been launched earlier this year.

Upgoing developments for PQC

- NIST PQC standards should be ready by 2024.
- Integration and deployment will probably take another 5-10 years
- Some candidates have already been chosen:
  - 1 lattice based KEM
  - 2 lattice based signatures
  - 1 hash based signature
- Fourth (last) round only features code based candidates
- A call for additional signatures has been launched earlier this year.

Thanks!

Références

Nicolas Aragon, Olivier Blazy, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor.
Durandal: a rank metric based signature scheme.
Cryptology ePrint Archive, Report 2018/1192, 2018.
https://eprint.iacr.org/2018/1192.

Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg.
On the inherent intractability of certain coding problems (corresp.).
IEEE Trans. Information Theory, 24(3):384–386, 1978.

Whitfield Diffie and Martin Hellman.
New directions in cryptography.
IEEE transactions on Information Theory, 22(6):644–654, 1976.

Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich.
Wave: A new code-based signature scheme.
Cryptology ePrint Archive, Report 2018/996, 2018.
https://eprint.iacr.org/2018/996.

Robert J. McEliece.
A public key cryptosystem based on algebraic coding theory.
In *Jet Propulsion Laboratory DSN Progress Report*, pages 42–44, 1978.

Ronald L Rivest, Adi Shamir, and Leonard Adleman.
A method for obtaining digital signatures and public-key cryptosystems.
Communications of the ACM, 21(2):120–126, 1978.

Peter W. Shor.
Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.
SIAM J. Comput., 26(5):1484–1509, 1997.