Algo de Grover : n entrées non trié, on recherche en $O(\sqrt{n})$ au lieu de $O(n)$

↳ conséquence : il faut doubler la taille de la clé (pour la même sécu) et +33% sur les fonctions de hachage (pour la même sécu)

Algo de Shor : • meilleur algo $(1.9(\log N)^{1/3}(\log\log N)^{2/3})$ en classique

• en quantique c'est de l'ordre du cubique.

↳ conséquence : fin de la crypto asymétrique (RSA, DSA, ECDSA, Elgamal) => plus d'échange de clés

$\mathbb{Z}_p$: entier modulo p ($= \mathbb{Z}/p\mathbb{Z}$)

$\mathbb{F}_q$ existe (corps fini modulo q) $\Leftrightarrow q = p^r$ pour p premier et $r \geq 1$

inverse de 7[11] ? pgcd(7,11) = 1 => ∃ un inverse d'après le thm de Bézout → algo d'Euclide :

$11 = 7\times1 + 4 \rightarrow 4 = 11 - 7\times1$

$7 = 4\times1 + 3 \rightarrow 3 = 7 - 4\times1$

$4 = 3\times1 + 1 \rightarrow 1 = 4 - 3\times1$

$\rightarrow 1 = 4 - (7 - 4\times1) = 4\times2 - 7 = \ldots = 11\times2 - 7\times3$

on s'arrête au reste = 1

$\Rightarrow 1 \equiv -7\times3\ [11]$

$\Rightarrow -3 = 8\ [11]$ est l'inverse

p premier et grand, g générateur $(\mathbb{Z}/p\mathbb{Z})^*$

- (g,p) = (2,7) ? $g^1=2, g^2=4, g^3=1, g^4=2, g^5=4, g^6=1$ ✗ n'engendre pas $\mathbb{Z}/p\mathbb{Z}$

→ avec g = 3 engendre $\mathbb{Z}/7\mathbb{Z}$

Savoir si on engendre $\mathbb{Z}/p\mathbb{Z}$: $g^{\frac{p-1}{2}}[p] = \begin{cases} +1 \Rightarrow g \text{ n'engendre pas} \\ -1 \Rightarrow g \text{ engendre } \mathbb{Z}/p\mathbb{Z} \end{cases}$

PQC : error correcting codes, lattices, multivariante, hash function, elliptic curves isogenies

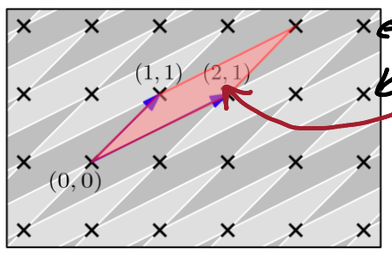
Rappel :

- norme : $\|x\| \geq 0\ \forall x$, $\|x\| = 0 \Leftrightarrow x = 0$, $\|x+y\| \leq \|x\| + \|y\|\ \forall x, y$
- $\vec{u}, \vec{v}$ linéaire indépendant $\Leftrightarrow (\alpha\vec{u} + \beta\vec{v} = \vec{0} \Leftrightarrow \alpha = \beta = 0)$
- $\|x\|_\infty = \sup\{x[i], \forall i \in \text{Domain}(x)\}$
- $u * v = (1,0,1,1) * (0,1,1,0) = (1+x^2+x^3)(x+x^2) = x^2 + x^3$ produit de convolution

↳ dans $\mathbb{F}_2^4 \simeq \mathbb{F}_2[x]/(x^4-1)$ $\Rightarrow x^4 - 1 = 0 \Rightarrow x^4 = 1$

Lattice : réseau Euclidien → r2o de dimension n et est un sous-groupe de $\mathbb{R}^n$

$(b_1, \ldots, b_n) \in \mathbb{R}^n$, $A(b_1, \ldots, b_n) : A = \{\sum_{i=1}^{n} x_i b_i ; x_i \in \mathbb{Z}\} \subset \mathbb{R}^n$



← solutions générés par la base

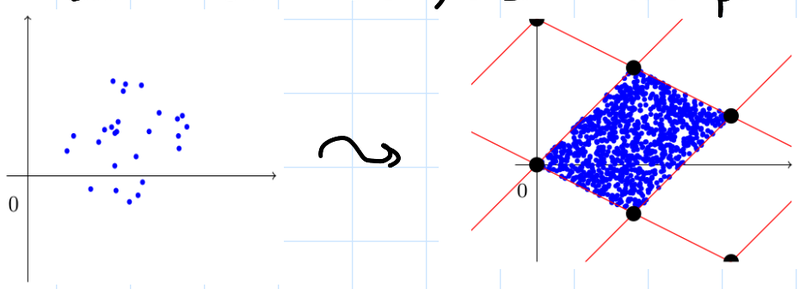
→ parallélépipède fondamentale

→ distance minimale du réseau L : $\lambda_1(L)$:

$\lambda_1(L) = \inf\{\|x-y\|, x \neq y \in L\}$

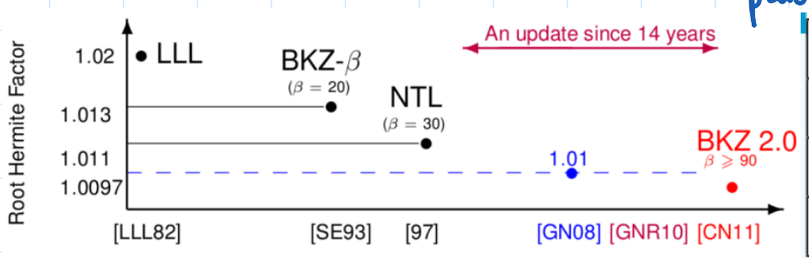
"on va utiliser la bonne base en clé privé, la mauvaise en clé publique mais elle sera assez proche donc décodable par clé privé"

→ sans la bonne base, c'est un hard-problem



si on utilise trop la même base, on leak le parallélépipède fondamentale !!

Facteur d'Hermite : $\delta = \left(\frac{\lambda_1}{\det L^{1/n}}\right)^{1/n}$ → plus il est petit (~1), plus l'attaque est efficace



k	δ
80	1.00783
100	1.00696
128	1.00602

Cryptosystem	Category	Key Size	Security Parameter	Quantum Algorithm Expected to Defeat Cryptosystem	# Logical Qubits Required	# Physical Qubits Required[a]	Time Required to Break System[b]	Quantum-Resilient Replacement Strategies
AES-GCM[c]	Symmetric encryption	128 192 256	128 192 256	Grover's algorithm	2,953 4,449 6,681	4.61×10^6 1.68×10^7 3.36×10^7	2.61×10^{12} years 1.97×10^{22} years 2.29×10^{32} years	
RSA[d]	Asymmetric encryption	1024 2048 4096	80 112 128	Shor's algorithm	2,050 4,098 8,194	8.05×10^6 8.56×10^6 1.12×10^7	3.58 hours 28.63 hours 229 hours	Move to NIST-selected PQC algorithm when available
ECC Discrete-log problem[e-g]	Asymmetric encryption	256 384 521	128 192 256	Shor's algorithm	2,330 3,484 4,719	8.56×10^6 9.05×10^6 1.13×10^6	10.5 hours 37.67 hours 55 hours	Move to NIST-selected PQC algorithm when available
SHA256[h]	Bitcoin mining	N/A	72	Grover's Algorithm	2,403	2.23×10^6	1.8×10^4 years	
PBKDF2 with 10,000 iterations[i]	Password hashing	N/A	66	Grover's algorithm	2,403	2.23×10^6	2.3×10^7 years	Move away from password-based authentication

Passe à ~18 ~20

Best know attacks : Lattice Reduction resulting basis B' is not that good

↳ LLL algorithm : polynomial-time algorithm, but exp approximation factor

LBC : encryption : n, m, q, α ← on génère une base de norme petite (erreur)

Keygen : $e \leftarrow \{-1,0,1\}^n \rightarrow s_k \in \{-1,0,1\}^n$ et $p_k = (A, b)$ avec $b = As + e$

Encrypt : $r \leftarrow \{0,1\}^n \rightarrow u = r^T A$ et $v = r^T b + \lfloor \frac{q}{2} \rfloor \times m$

Decrypt : compute $f = v - u^T s \rightarrow$ si f est proche de 0 : output = 0 sinon : 1

$\rightsquigarrow f = r^T b + \frac{q}{2}m - r^T A s = r^T(As+e) + \frac{q}{2}m - r^T A s = r^T A s + r^T e + \frac{q}{2}m - r^T A s$

NTRU Sign (Lattice-based signature) :

$f, g = \begin{cases} d \text{ coefficients } +1 \\ N-d \text{ coefficients } 0 \end{cases}$ $\quad fG + qf^* G - F^* g = q$

$h = g^* f^{-1} \xleftarrow{s} R_q = \mathbb{Z}_q[X]/(X^N+1)$

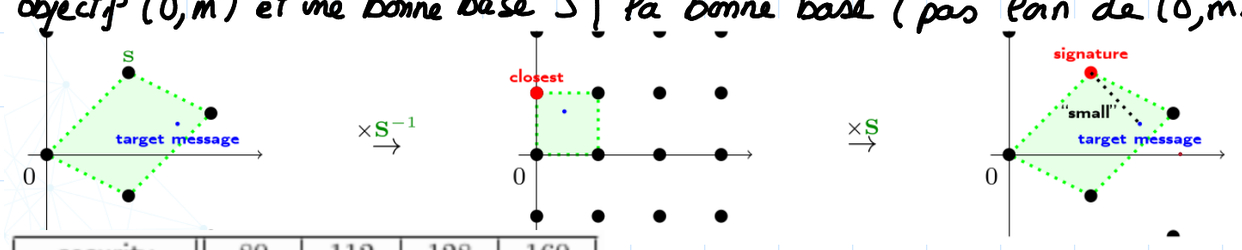
$P = \left(\begin{array}{c|c} 1 & h \\ \hline 0 & q \end{array}\right)$ $\quad S = \left(\begin{array}{c|c} f & g \\ \hline F & G \end{array}\right)$ $\rightsquigarrow A_{h,q} = \{(u, u^* h\ [q]), u \in R_q\}$

Sign : $\mu \in \{0,1\}^*$

$m = H(\mu)$, résoudre CVP avec objectif (0,m) et une bonne base S

Verify : signature s

il faut vérifier le point de r2o avec la bonne base (pas loin de (0,m))



security	80	112	128	160
NTRUSign	1256	1576	1784	2367
$ECDSA_{sign}$	320	448	512	640
RSA	1024	2048	3072	4096

Taille des signatures en bits

↝ problem : si on signe trop

Secure lattice based signature [Lyu12] :

Keygen : $s_k : S \xleftarrow{\$} [-d, d]^{m \times k}$, $P_k : A \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ et $T = A.S \in \mathbb{Z}_q^{n \times k}$

Sign : 1ère étape (trouver pre-image) : on calcule Sc la pre-image de Tc

2ème étape (cacher la pre-image) : on ajoute un bruit Gaussien y à Sc

Verify : (z, c). Il faut vérifier :

- $H(Az - Tc, \mu) = c$ → c'est un lattice vector
- $\|z\| \leq^* n\sigma\sqrt{m}$ → a une norme raisonable

*$A(Sc+y) - A\,Sc$

100 bits of security

n	512	512	512	512	512
m	8,786	8,139	3,253	1,024	1,024
k	80	512	512	512	512
$\log_2(q)$	27	25	33	18	26
d	1	1	31	1	31
M (retries)	2.72	2.72	2.72	7.4	7.4
≈ sign size	163,000	142,300	73,000	14,500	19,500
≈ pk size	2^{20}	$2^{22.5}$	2^{23}	$2^{19.5}$	$2^{21.5}$
≈ sk size	2^{20}	$2^{22.5}$	2^{23}	$2^{22.1}$	$2^{22.7}$

CBC (Code Based Cryptography) → cf TD p.4 ↳ chiffrement

↳ très efficace : algèbre linéaire, arithmétique simple (modulo 2 vs modulo $2^{\sim 1024}$ pour RSA), hautement parallélisable

↳ Taille de clé conséquentes (quasi-cyclique), hypothèse d'indistingabilité de la famille de code utilisé (+ technique)

mots du code : $\mathcal{C} = \{x \times G, x \in \mathbb{F}_2^k\} = \{s \in \mathbb{F}_2^n \text{ tq } Hs^T = 0\}$

Quasi-cyclic Moderate Density Parity-Check Codes

↳ on peut créer des clés en ajoutant des erreurs et corriger les erreurs pour retrouver les clés par algo itératif (bitflipping)

CBC – Signature efficace :

s_k : x de poid faible w, p_k : H
m : message et y de poid faible
$c = \mathcal{H}(Hy^T, m)$ de poid faible
$z = x \cdot c + y$

z, c →

$wt(z) \leq \tilde{w}$ pas trop grand
Vérifier que $\mathcal{H}(Hz^T - s \cdot c, m) == c$
en pratique c'est bof...

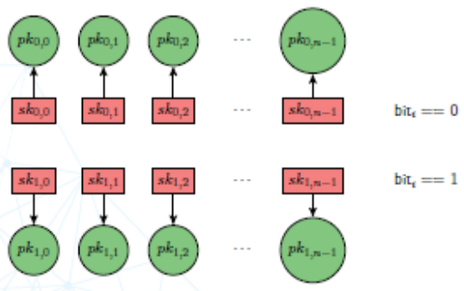
Claimed security	Persichetti's OTS parameters n	w_1	w_2	δ	xBF parameters τ	N	Verification t_{verify} (ms)	Cryptanalysis t_{break} (ms)
80	4801	90	100	10	7	5	22.569	165.459
	3072	85	85	7	5	5	14.271	68.858
128	9857	150	200	12	9	10	99.492	453.680
	6272	125	125	10	7	10	42.957	288.442

Rappels hash : collision resistance hard ⇒ 2nd pre-image hard
⇏ 1st pre image hard

Lamport One-Time Signature scheme :

Assume you want to sign a n-bits long message m with CRHF $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$.

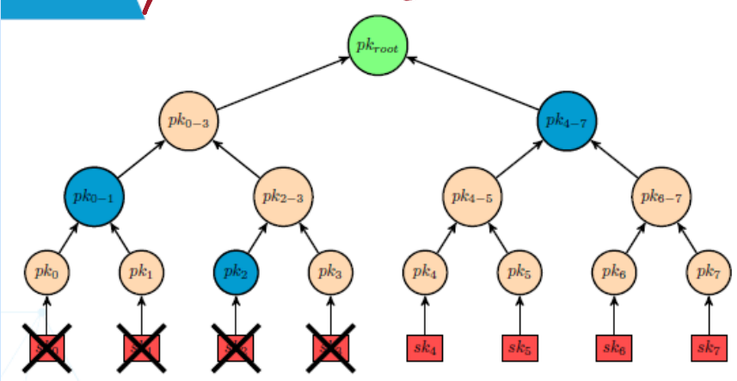
- Generate $2n$ random values $sk_{0,i}, sk_{1,i}$, for $i \in \{0, n-1\}$
- Compute the associated public key: $pk = \{pk_{b,i}\}_{\substack{i \in \{0,n-1\} \\ b \in \{0,1\}}} = \{\mathcal{H}(sk_{b,i})\}$



- Assume $m =$ | 1 | 1 | 0 | ⋯ | 1 |
- Then $\sigma =$ | $sk_{1,0}$ | $sk_{1,1}$ | $sk_{0,2}$ | ⋯ | $sk_{1,n-1}$ |
- Verify that $\mathcal{H}(\sigma_i) == pk_{m_i,i}$

Once you have signed one message, destroy the key.

Multiple-time signatures : Merkle approach



- public key
- intermediate signature hashes
- secret OTS keys
- revealed values for signature i

Assume you want to sign your 4th message (index 3 so).
Then reveal the signature using sk_3 along with the path to pk_{root}:

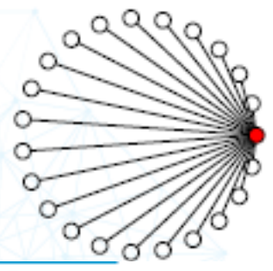
- pk_2
- pk_{0-1}
- pk_{4-7}

Don't forget to update the state: sk_3 burst!

23 people: explanation

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

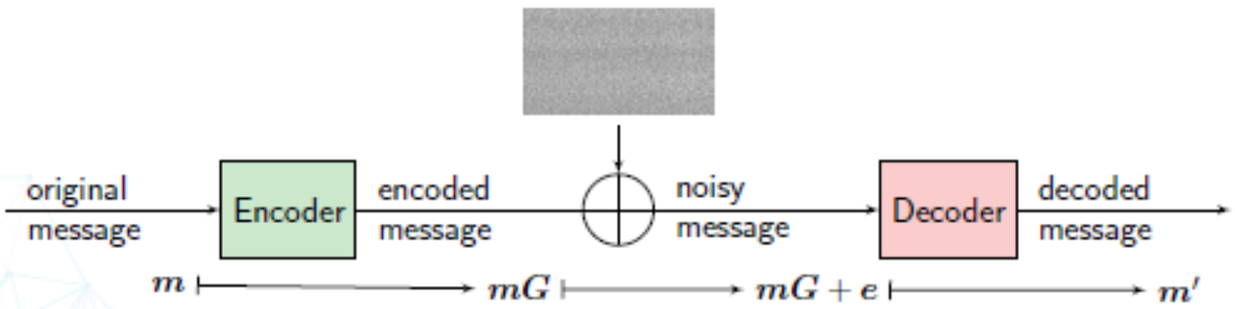
Probability $\mathbb{P}$ that there is no collision :



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

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

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.

Conclu du cours :

- 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.

Correction Code TD $G = (Id \quad A) \Rightarrow$ G systématique ($G \in \mathbb{F}_2^{4\times 7}$ par ex)

k = longueur = 7, n = dimension = 4, possède 2^n mots du code
distance minimale = plus petite distance de Hamming entre les 2^4 mots
$d = k - n \Rightarrow d - 1$ erreurs détectable, $\frac{d-1}{2}$ erreurs corrigeable

$$H^T = \begin{pmatrix} A \\ Id \end{pmatrix}$$ si G systématique.

cas simple : $c = \underbrace{mG}_{c_0} + e$

$Hc^T = H(c_0 \oplus e)^T = Hc_0^T \oplus He^T = H(mG)^T \oplus He^T = \cancel{HG^T m^T} \oplus He^T$ car $HG^T = 0$ par constru.

on calcule $s^T = Hc^T$, on cherche s^T dans H et la ligne nous donne $\tilde{e}$ l'index/position du bitflip. On le corrige.

↳ **McEliece** $\tilde{G} = SGP$, S aléatoire inversible ($\det S \neq 0$), P permutation (1 seul '1' par ligne et par colonne $\Rightarrow P^{-1} = P^T$)
$c = m\tilde{G} \oplus e$. Pour déchiffrer : On calcule $y = cP^{-1} = mSG \oplus \tilde{e}P^{-1}$.
Après correction on a mSG : grâce à G on retrouve mS $\Rightarrow m = mSS^{-1}$

GGH $cB^{-1} = (mB' + e)B^{-1} = mUBB^{-1} + eB^{-1} = mU + eB^{-1} \in \mathbb{Q}^n$

Pour décoder il faut que eB^{-1} soit petit. → formellement $\|eB^{-1}\|_\infty < \frac{1}{2}$

$\lfloor cB^{-1} \rceil \times U^{-1} = mUU^{-1} = m$

- Decrypt $(c, U^{-1}, B^{-1}) = \lfloor cB^{-1} \rceil \times U^{-1}$
- Encrypt $(m, B') = mB' + e$

$pk = B', sk = (U^{-1}, B^{-1}), B' = UB$

Algorithm 1: KeyGen
Input: Paramètres k, n, M, et σ
Output: pk = B′, sk = (B^{-1}, U^{-1})
1 **while** $\det(B) = 0$ **do**
2 $\quad B \xleftarrow{\$} \{-M, \ldots, M\}^{n \times n}$
3 **while** $|\det(U)| \neq 1$ **do**
4 $\quad U \xleftarrow{\$} \{-M, \ldots, M\}^{n \times n}$
5 **return** $pk = B'$, $sk = (B^{-1}, U^{-1})$;

Algorithm 2: Encrypt
Input: Message m à chiffrer, clé publique pk = B′
Output: Chiffré c
1 $e \xleftarrow{\$} \{-\sigma, \ldots, \sigma\}^n$;
2 **return** $c = mB' + e$;

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \Rightarrow A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

$$B = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$B^{-1} = \frac{1}{|B|} \begin{bmatrix} (ei-fh) & -(bi-ch) & (bf-ce) \\ -(di-fg) & -(ai-cg) & -(af-cd) \\ (dh-eg) & -(ah-bg) & (ae-bd) \end{bmatrix}$$