

Petit panorama de la Cryptographie Post-Quantique (PQC)

Alexandre Wallet

Ecole Polytechnique
06/03/2023

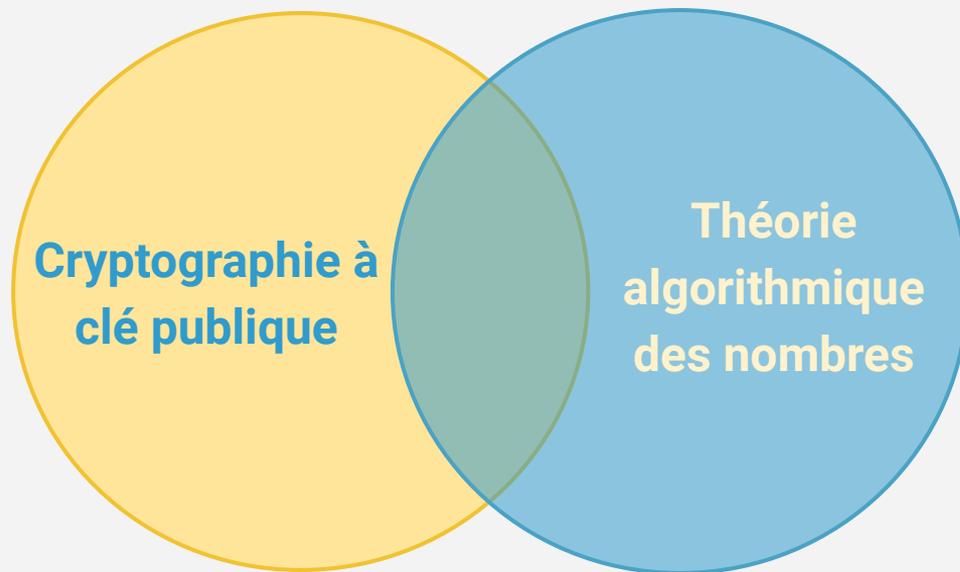
“C ki ? Y fé koi ?”

Chargé de recherche (CR) Inria, à l'IRISA, Rennes

équipe CAPSULE : <https://team.inria.fr/capsule/>

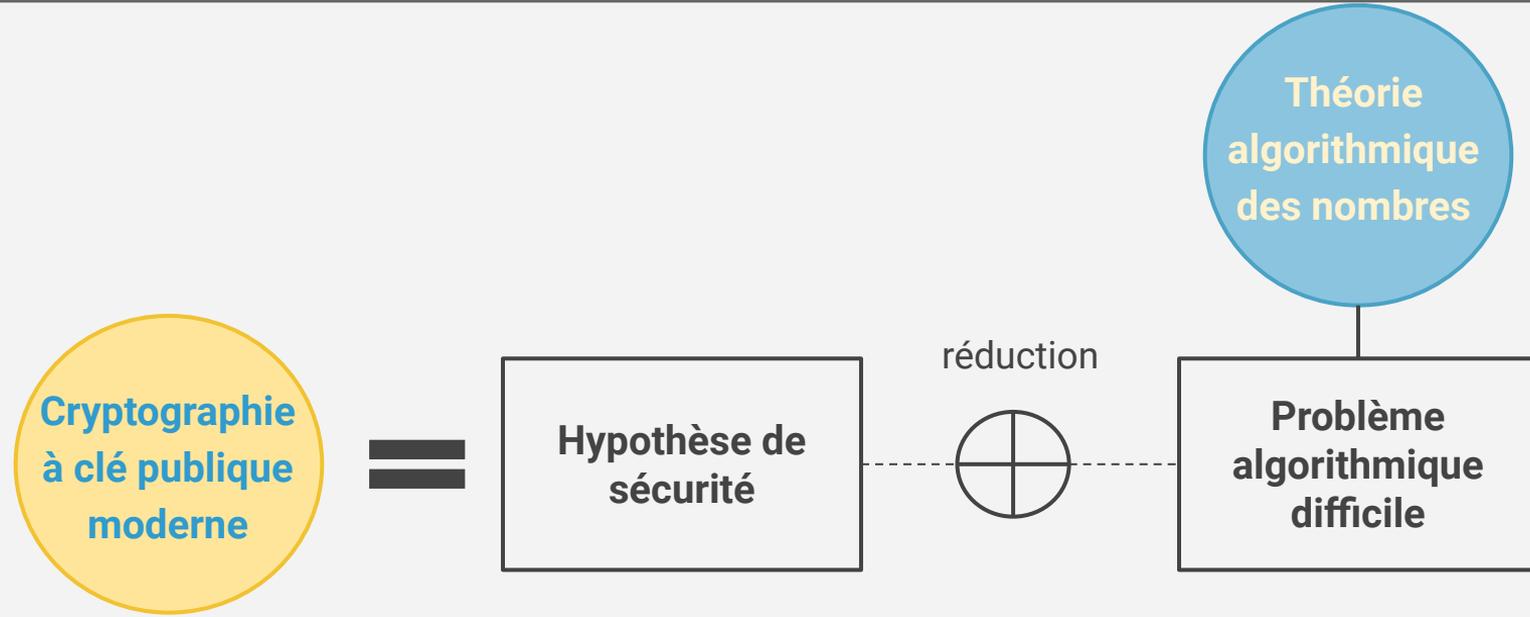
Web : <https://awallet.github.io>

Mail : alexandre.wallet@inria.fr



Ma recherche :
Evaluation, déploiement de la cryptographie post-quantique

Cryptographie moderne



Cryptographie moderne

Réduction $A \longrightarrow B$: “si je sais résoudre B , je sais résoudre A .” (en temps polynomial)

ex: résoudre un système linéaire \longrightarrow pivot de Gauss

Théorie
algorithmique
des nombres

Cryptographie
à clé publique
moderne

=

Hypothèse de
sécurité

réduction



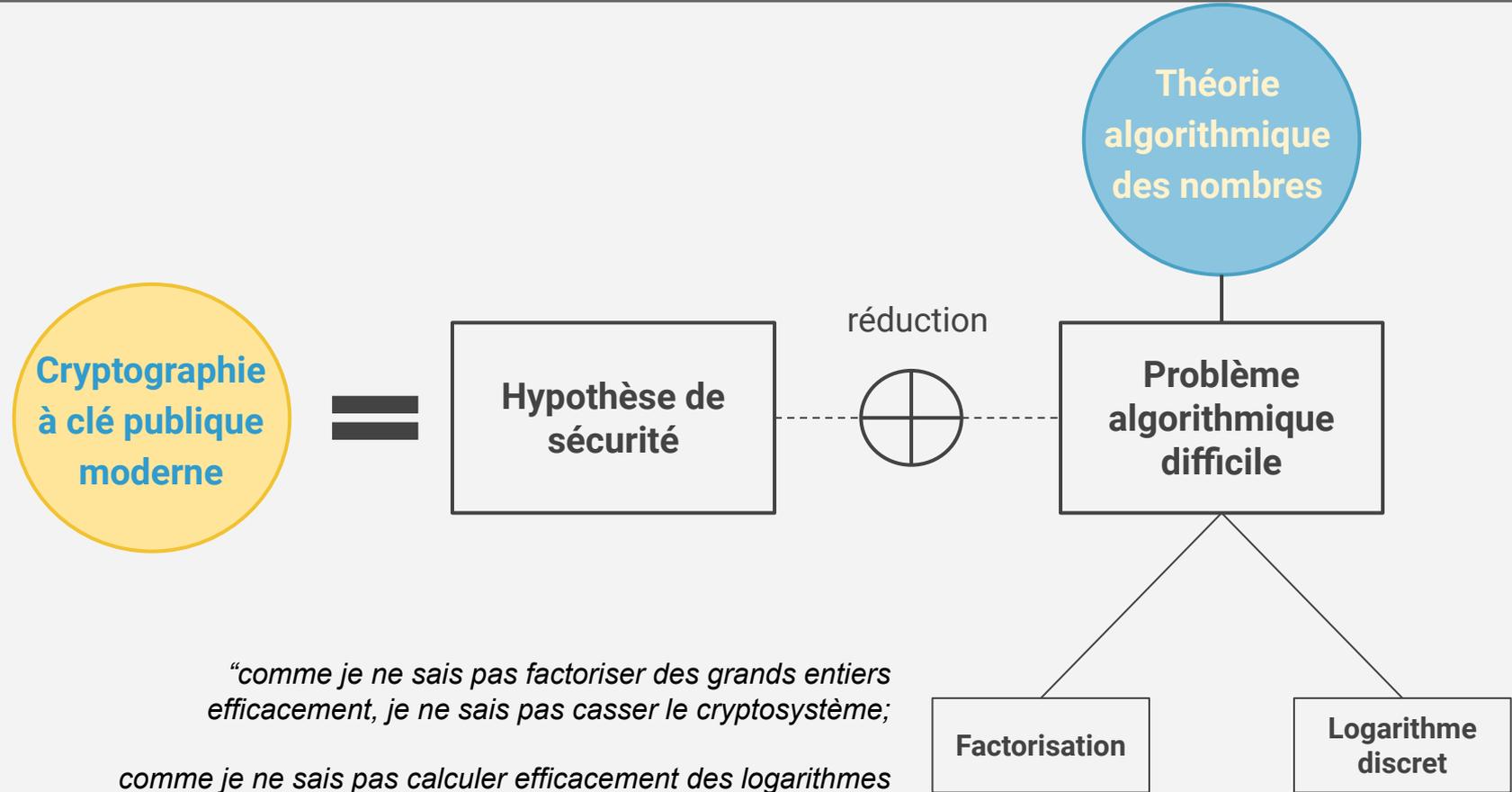
Problème
algorithmique
difficile

“casser le schéma”

Difficile = complexité exponentielle

“Facile” = complexité polynomiale

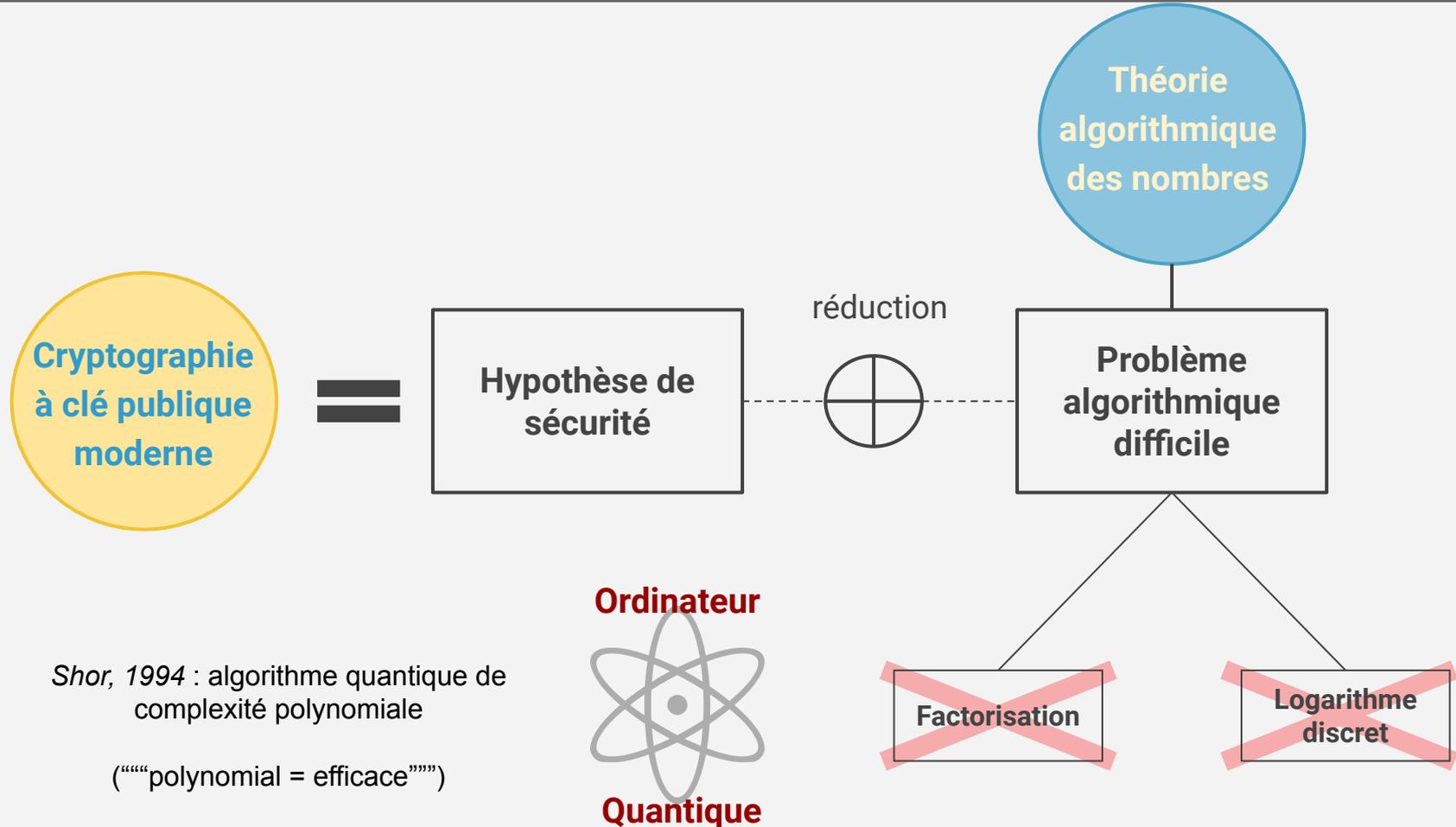
Cryptographie moderne



“comme je ne sais pas factoriser des grands entiers efficacement, je ne sais pas casser le cryptosystème;

comme je ne sais pas calculer efficacement des logarithmes discrets, je ne sais pas casser le cryptosystème.”

Cryptographie moderne



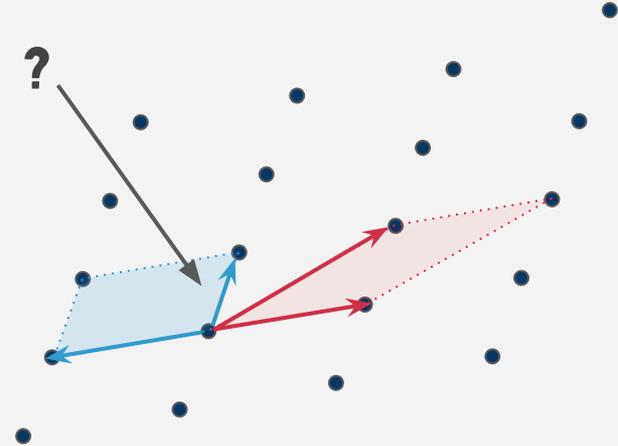
“Post-quantique ?”

**Cryptographie
post-quantique**



Cryptographie reposant sur des problèmes algorithmiques difficiles même pour un ordinateur quantique

Ex: Calculer des vecteurs courts dans des réseaux euclidiens



Contexte actuel

Appel à standardisation post-quantique du NIST : effort global pour concevoir des échanges de clés (KEM) et des signatures devant résister aux algorithmes quantiques.

Novembre 2017

Mai 2022

Round 1

Round 2

Round 3

~70 propositions :

- réseaux euclidiens
- codes correcteurs
- isogénies
- systèmes multivariés
- autres...

26 propositions :

- 17 KEMs
- 9 signatures

7 finalistes : 4 KEMs, 3 signatures
+8 alternatives

Contexte actuel

Appel à standardisation post-quantique du NIST : effort global pour concevoir des échanges de clés (KEM) et des signatures devant résister aux algorithmes quantiques.

Plus récemment :

- 2021 : une des signatures finalistes est quasi-cassée
- Mai 2022 : annonce des vainqueurs et des “autres”
- Juillet 2022 : un des “autres” KEMs complètement cassé
- Septembre 2022 : pas assez de diversité en signature
=> nouvel appel (si possible sans réseaux :)

Les grandes étapes de la conception

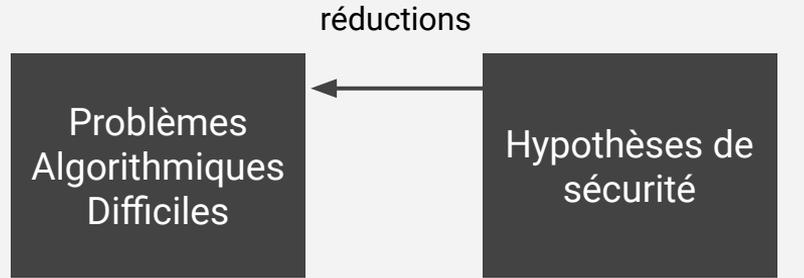
Hypothèses de
sécurité

*“Chiffrer/signer doit
être dur sans sk ”*

y arriver => résoudre
un problème difficile(?)



Les grandes étapes de la conception



problèmes difficiles
sur une structure
mathématique

Lesquels ?
Durs comment ?



*"Chiffrer/signer doit
être dur sans sk"*

y arriver => résoudre
un problème difficile(?)



Les grandes étapes de la conception

Structures maths : factoriser, log discret (classique)
codes correcteurs, réseaux euclidiens, systèmes polynomiaux, isogénies, ...

réductions

Problèmes
Algorithmiques
Difficiles

Hypothèses de
sécurité

problèmes difficiles
sur une structure
mathématique

*“Chiffrer/signer doit
être dur sans sk”*

y arriver => résoudre
un problème difficile(?)

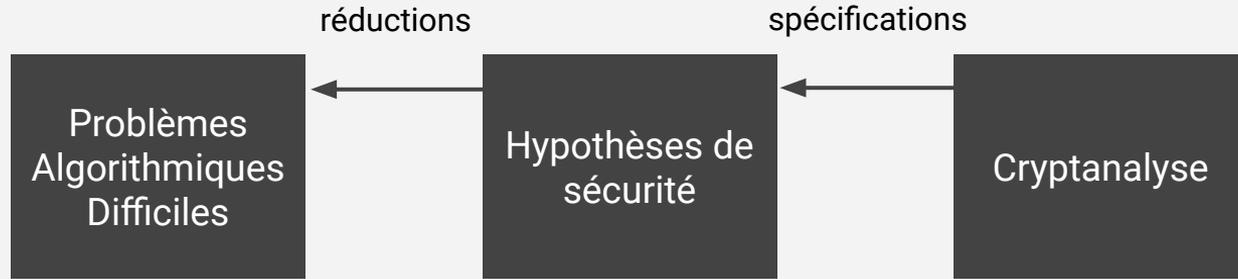
Lesquels ?
Durs comment ?



Les grandes étapes de la conception

Structures maths : factoriser, log discret (classique)

codes correcteurs, réseaux euclidiens, systèmes polynomiaux, isogénies, ...



problèmes difficiles
sur une structure
mathématique

Lesquels ?
Durs comment ?



*“Chiffrer/signer doit
être dur sans sk”*

y arriver => résoudre
un problème difficile(?)



Quelle dimension ?
Quels paramètres ?

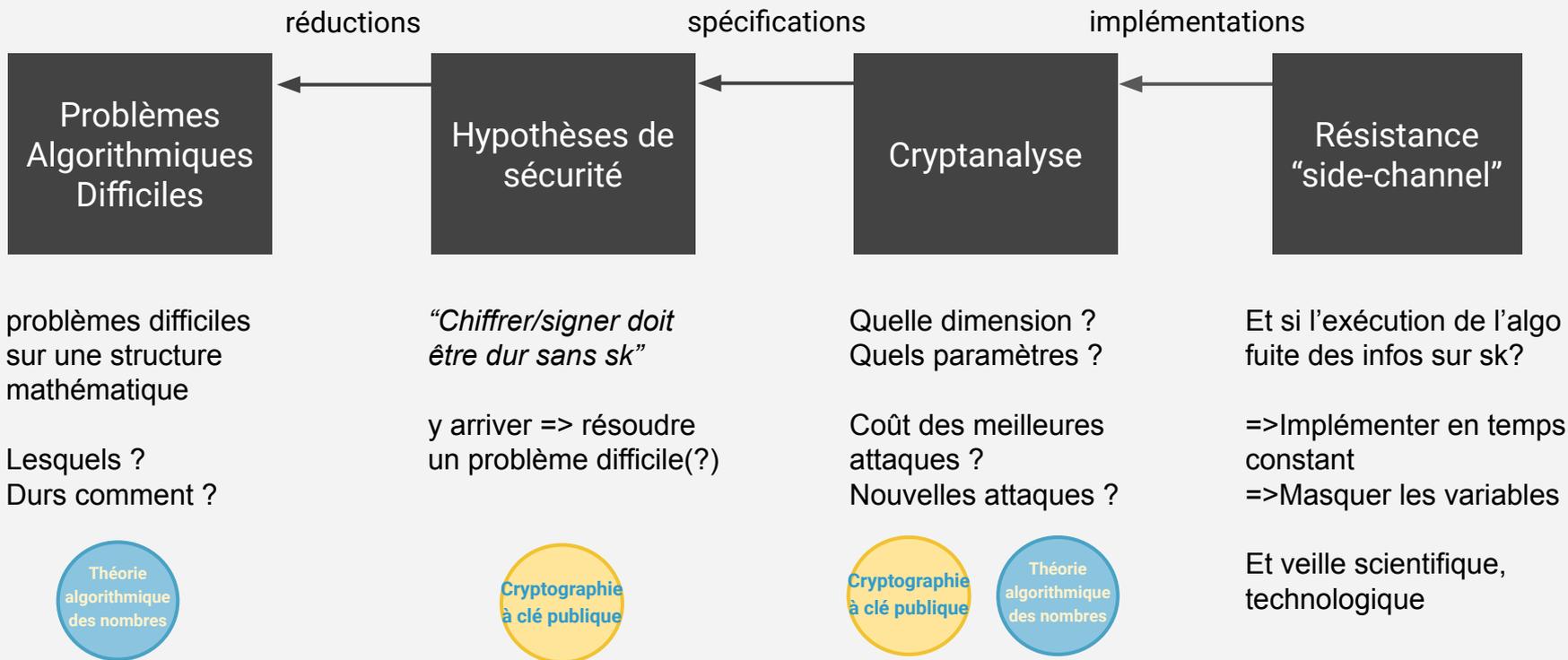
Coût des meilleures
attaques ?
Nouvelles attaques ?



Les grandes étapes de la conception

Structures maths : factoriser, log discret (classique)

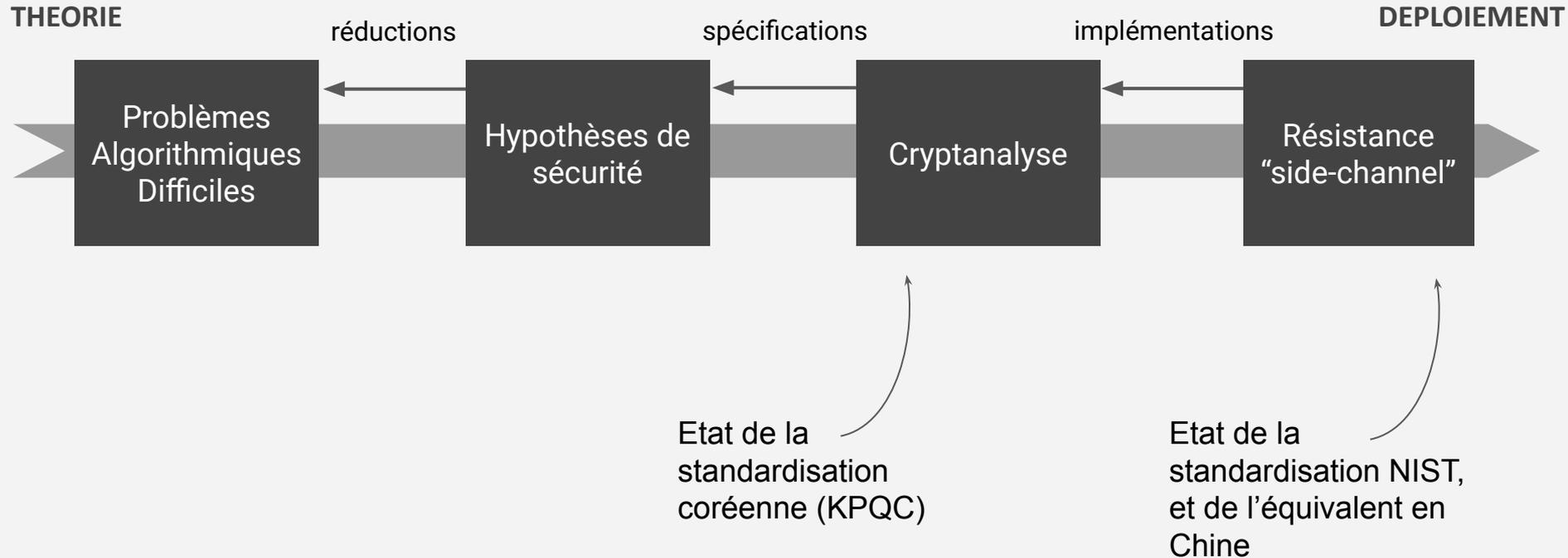
codes correcteurs, réseaux euclidiens, systèmes polynomiaux, isogénies, ...



Et maintenant ?

Structures maths : factoriser, log discret (classique)

codes correcteurs, réseaux euclidiens, systèmes polynomiaux, isogénies, ...



Et maintenant ?

réseaux euclidiens

THEORIE

DEPLOIEMENT

Cryptanalyse

Résistance
"side-channel"

VEILLE TECHNOLOGIQUE
pendant la durée de vie du schéma

Etat de la
standardisation
coréenne (KPQC)

Etat de la
standardisation NIST,
et de l'équivalent en
Chine

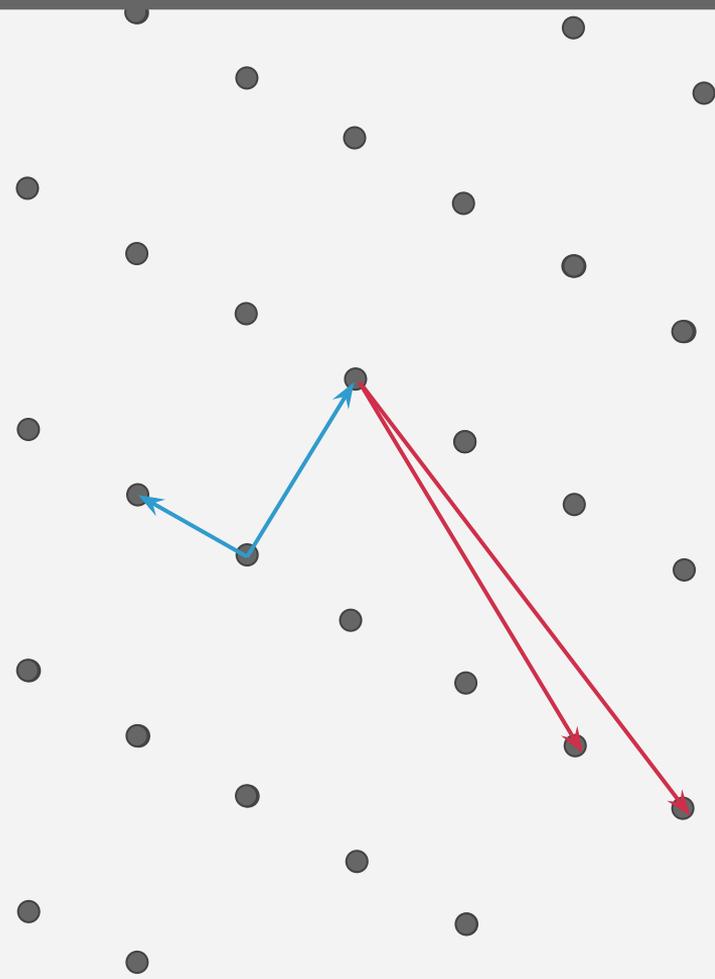
Réseaux euclidiens, partie 1

Réseaux euclidiens

Réseau euclidien \mathcal{L} : sous-groupe *discret* de \mathbb{R}^n

b_1, \dots, b_n linéairement indépendants tels que $\mathcal{L} = \bigoplus_i \mathbb{Z}b_i$
est une *base* de \mathcal{L} . n est le rang de \mathcal{L}

Si \mathbf{B} est la matrice des b_i , on note aussi $\mathbf{B}\mathbb{Z}^n$



Réseaux euclidiens

Réseau euclidien \mathcal{L} : sous-groupe *discret* de \mathbb{R}^n

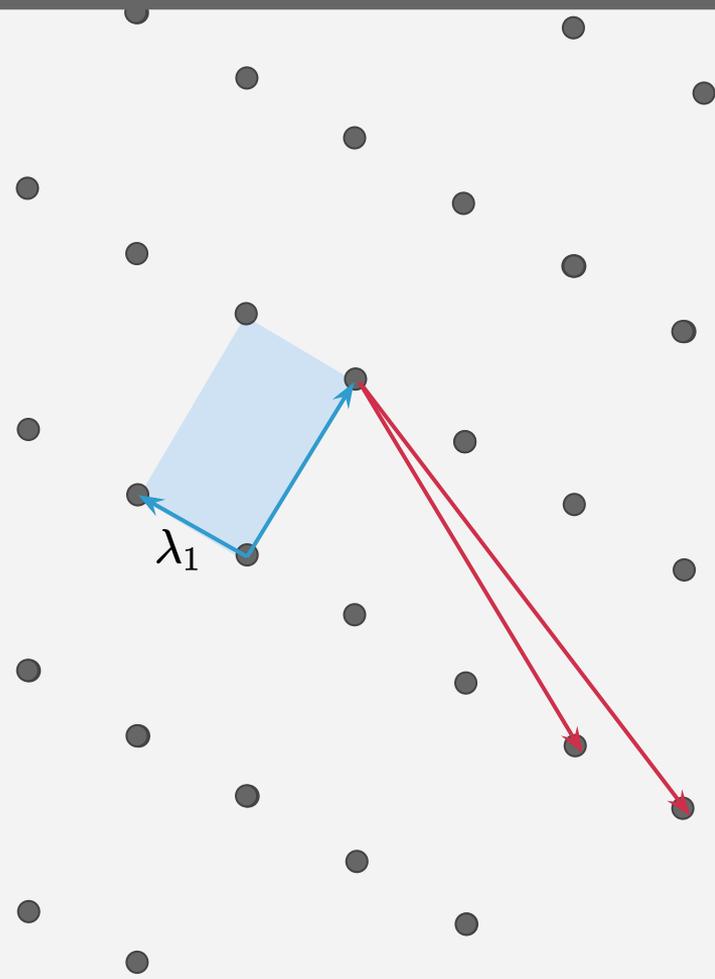
b_1, \dots, b_n linéairement indépendants tels que $\mathcal{L} = \bigoplus_i \mathbb{Z}b_i$
est une *base* de \mathcal{L} . n est le rang de \mathcal{L}

Si \mathbf{B} est la matrice des b_i , on note aussi $\mathbf{B}\mathbb{Z}^n$

Le *volume* de \mathcal{L} est le déterminant d'une de ses bases.

La longueur d'un plus court vecteur est notée λ_1

Théorème (Minkowski): $\lambda_1 \leq \sqrt{n}(\det \mathcal{L})^{1/n}$

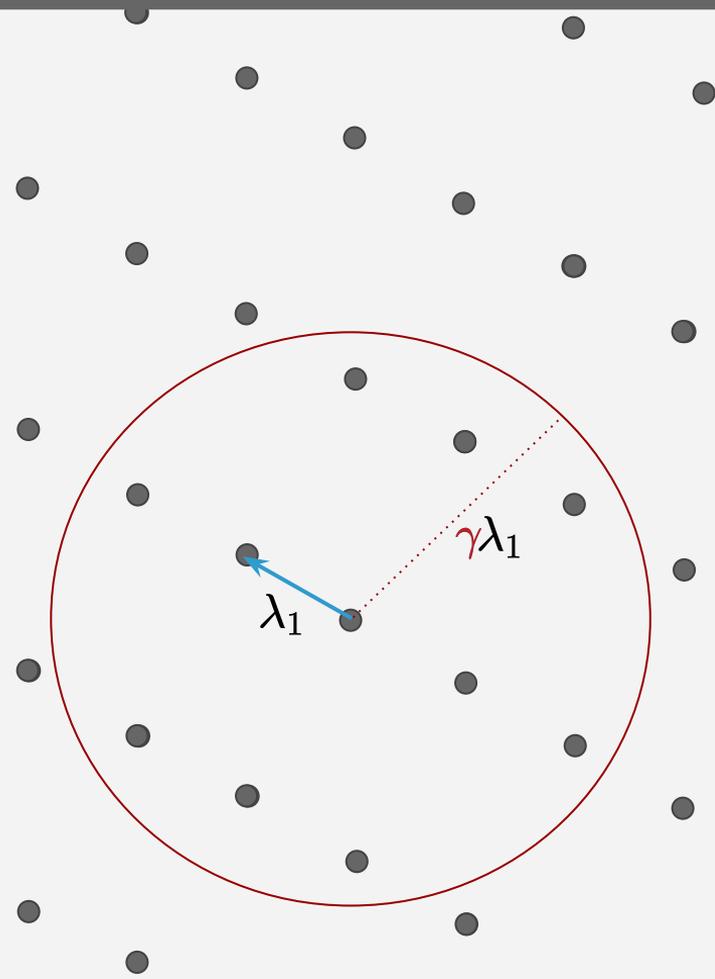


Shortest Vector Problem (SVP)

SVP : calculer $v \in \mathcal{L}$ tel que $\|v\| = \lambda_1$

SVP _{γ} : Pour $\gamma \geq 1$, calculer $v \in \mathcal{L}$ tel que $\|v\| \leq \gamma \lambda_1$

Meilleur algo pour résoudre **SVP_{poly(n)}** : temps $2^{O(n)}$



Shortest Vector Problem (SVP)

SVP : calculer $v \in \mathcal{L}$ tel que $\|v\| = \lambda_1$

SVP _{γ} : Pour $\gamma \geq 1$, calculer $v \in \mathcal{L}$ tel que $\|v\| \leq \gamma \lambda_1$

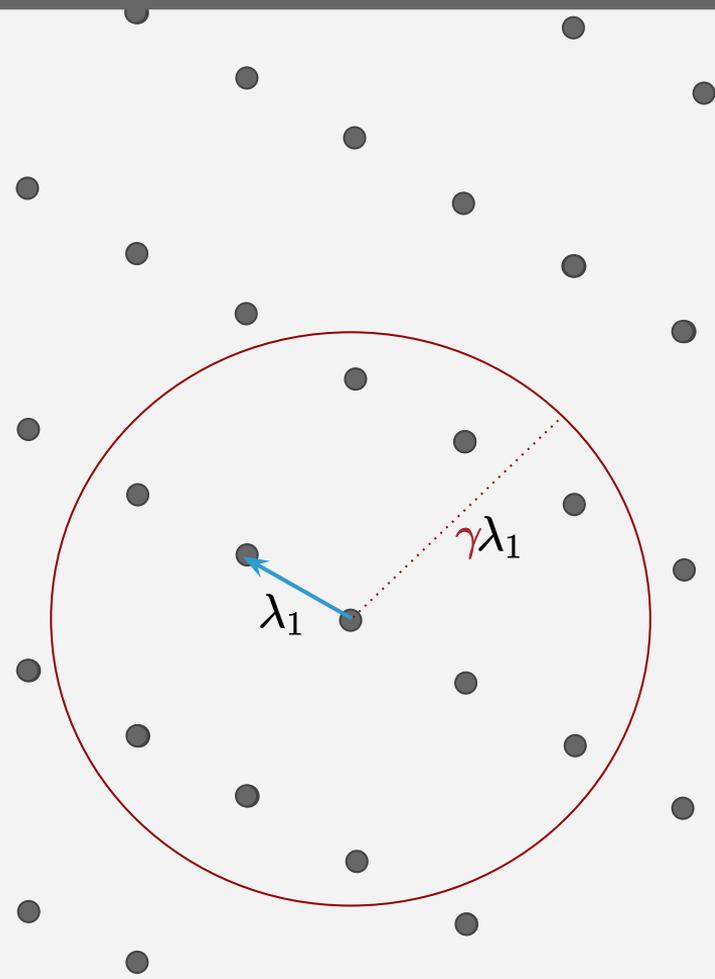
Meilleur algo pour résoudre **SVP_{poly(n)}** : temps $2^{O(n)}$

Comment calculer des vecteurs courts ?

Énumération, crible... au moins exponentiels.

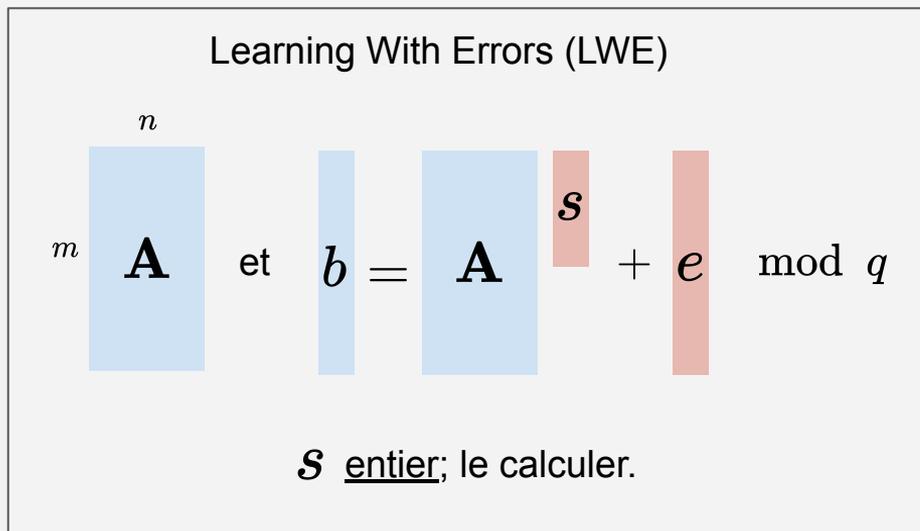
Algorithmes de réductions : LLL, BKZ, ... font des appels à l'énumération/crible en plus petite dimension

LLL est polynomial-time, mais les vecteurs trouvés sont beaucoup trop long **en grande dimension.**



Chiffrement reposant sur les réseaux euclidiens

Une hypothèse de sécurité en vogue



Typiquement :

- $\mathbf{A} \in \mathbb{Z}/q\mathbb{Z}^{m \times n}$ tiré uniformément
- $\mathbf{s} \in \mathbb{Z}_q^n$ tiré uniformément
- e est aléatoire entier et **petit**

(On utilise aussi les variantes décisionnelles)

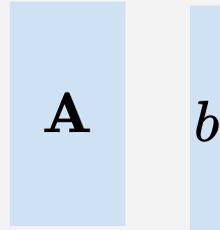
- Résoudre LWE \Rightarrow résoudre efficacement **SVP**_{poly(n)} (Regev, 2005)
- Réduction pire-cas moyen-cas : résoudre LWE en moyenne implique un algorithme efficace pour SVP dans **n'importe quel réseau** !
- Autres fonctionnalités : chiffrement homomorphe, fondé sur l'identité, sur les attributs, signatures...

Le chiffrement de Regev (2005)

Alice



s



Bob

veut chiffrer $\mu \in \{0, 1\}$

$$r \leftarrow^{\$} \{0, 1\}^m$$

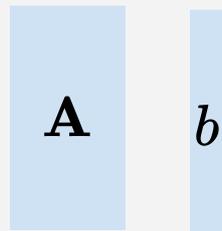
Le chiffrement de Regev (2005)

Alice



s

$$\begin{aligned}d &= b' - \langle \mathbf{a}', s \rangle \\ &= r^t e + \mu \frac{q}{2}\end{aligned}$$



(\mathbf{a}', b')

Bob

veut chiffrer $\mu \in \{0, 1\}$

$$r \leftarrow^{\$} \{0, 1\}^m$$

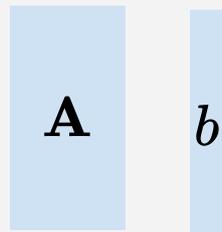
$$(\mathbf{a}', b') = (r^t \mathbf{A}, r^t b + \mu \frac{q}{2})$$

Le chiffrement de Regev (2005)

Alice

s

$$\begin{aligned}d &= b' - \langle \mathbf{a}', s \rangle \\ &= r^t e + \mu \frac{q}{2}\end{aligned}$$



(\mathbf{a}', b')

Bob

veut chiffrer $\mu \in \{0, 1\}$

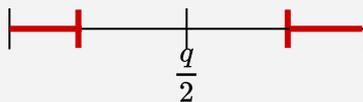
$$r \leftarrow^{\$} \{0, 1\}^m$$

$$(\mathbf{a}', b') = (r^t \mathbf{A}, r^t b + \mu \frac{q}{2})$$

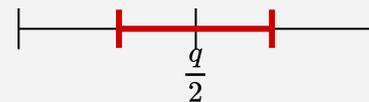
$$\text{Dec}(\mathbf{a}', b') = \begin{cases} 0 & \text{si } d \approx 0 \\ 1 & \text{si } d \approx \frac{q}{2} \end{cases}$$

Correct si $r^t e \leq \frac{q}{4}$:

$\mu = 0$



$\mu = 1$



De Regev au futur standard

Limites du chiffrement de Regev textbook:

1. chiffre bit par bit
2. un peu rigide
3. en pratique, grandes matrices (> 500 lignes colonnes) : **clés trop grosses, opérations “lentes”**
4. chiffrer en asymétrique ??!

$\log q \approx 12$: la matrice
publique prend
~ 800 KBytes !

algèbre linéaire au
moins **quadratique**

De Regev au futur standard

Limites du chiffrement de Regev textbook:

1. chiffre bit par bit
2. un peu rigide
3. en pratique, grandes matrices (> 500 lignes colonnes) : **clés trop grosses, opérations “lentes”**
4. chiffrer en asymétrique ??!

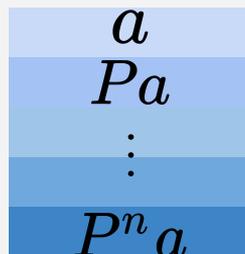
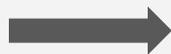
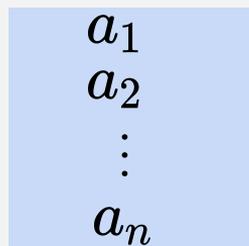
Solutions:

pour 1. et 3. : **utiliser des réseaux algébriquement structurés** (et des techniques supplémentaires)

pour 2. : variante dual-Regev (idées très similaires, repose sur LWE aussi)

pour 4. : transformation générique du chiffrement en mécanisme d'**encapsulation de clé (KEM)**

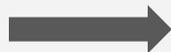
Réseaux algébriquement structurés ?



(En pratique on utilise même des versions blocs-structurées: chaque bloc est donnée par les permutations d'une ligne)

P permutation cyclique
(par exemple)

LWE



Struct-LWE

SVP



Struct-SVP



Réduction pire-cas moyen-cas pour les réseaux structurés correspondants

CRYSTALS-KYBER

Le KEM prioritaire du NIST

Crystals-Kyber : <https://pq-crystals.org/kyber/>

Structure algébrique : corps cyclotomiques

KEM prioritaire pour standardisation NIST

(NB : l'ANSSI ne recommande pas forcément Kyber par rapport à un autre concurrent, FrodoKEM)

Algorithm 5 KYBER.CPAPKE.Enc(pk, m, r): encryption

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n / 8 + 32}$

Input: Message $m \in \mathcal{B}^{32}$

Input: Random coins $r \in \mathcal{B}^{32}$

Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n / 8 + d_v \cdot n / 8}$

```
1:  $N := 0$ 
2:  $\hat{\mathbf{t}} := \text{Decode}_{12}(pk)$ 
3:  $\rho := pk + 12 \cdot k \cdot n / 8$ 
4: for  $i$  from 0 to  $k - 1$  do
5:   for  $j$  from 0 to  $k - 1$  do
6:      $\hat{\mathbf{A}}^T[i][j] := \text{Parse}(\text{XOF}(\rho, i, j))$ 
7:   end for
8: end for
9: for  $i$  from 0 to  $k - 1$  do
10:   $\mathbf{r}[i] := \text{CBD}_{\eta_1}(\text{PRF}(r, N))$ 
11:   $N := N + 1$ 
12: end for
13: for  $i$  from 0 to  $k - 1$  do
14:   $\mathbf{e}_1[i] := \text{CBD}_{\eta_2}(\text{PRF}(r, N))$ 
15:   $N := N + 1$ 
16: end for
17:  $\mathbf{e}_2 := \text{CBD}_{\eta_2}(\text{PRF}(r, N))$ 
18:  $\hat{\mathbf{r}} := \text{NTT}(\mathbf{r})$ 
19:  $\mathbf{u} := \text{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$ 
20:  $\mathbf{v} := \text{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_2 + \text{Decompress}_q(\text{Decode}_1(m), 1)$ 
21:  $c_1 := \text{Encode}_{d_u}(\text{Compress}_q(\mathbf{u}, d_u))$ 
22:  $c_2 := \text{Encode}_{d_v}(\text{Compress}_q(\mathbf{v}, d_v))$ 
23: return  $c = (c_1 || c_2)$ 
```

Le KEM prioritaire du NIST

Crystals-Kyber : <https://pq-crystals.org/kyber/>

Structure algébrique : corps cyclotomiques

Recalcul de A via la seed du PRNG

CBD : échantillonneur binomial pour le bruit

Multiplications via NTT (analogue arithmétique de la FFT), complexité quasi-linéaire

Algorithm 5 KYBER.CPAPKE.Enc(pk, m, r): encryption

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8 + 32}$

Input: Message $m \in \mathcal{B}^{32}$

Input: Random coins $r \in \mathcal{B}^{32}$

Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$

```
1:  $N := 0$ 
2:  $\hat{\mathbf{t}} := \text{Decode}_{12}(pk)$ 
3:  $\rho := pk + 12 \cdot k \cdot n/8$ 
4: for  $i$  from 0 to  $k - 1$  do
5:   for  $j$  from 0 to  $k - 1$  do
6:      $\mathbf{A}^T[i][j] := \text{Parse}(\text{XOF}(\rho, i, j))$ 
7:   end for
8: end for
9: for  $i$  from 0 to  $k - 1$  do
10:   $\mathbf{r}[i] := \text{CBD}_{\eta_1}(\text{PRF}(r, N))$ 
11:   $N := N + 1$ 
12: end for
13: for  $i$  from 0 to  $k - 1$  do
14:   $\mathbf{e}_1[i] := \text{CBD}_{\eta_2}(\text{PRF}(r, N))$ 
15:   $N := N + 1$ 
16: end for
17:  $\mathbf{e}_2 := \text{CBD}_{\eta_2}(\text{PRF}(r, N))$ 
18:  $\hat{\mathbf{r}} := \text{NTT}(\mathbf{r})$ 
19:  $\mathbf{u} := \text{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$ 
20:  $\mathbf{v} := \text{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_2 + \text{Decompress}_q(\text{Decode}_1(m), 1)$ 
21:  $c_1 := \text{Encode}_{d_u}(\text{Compress}_q(\mathbf{u}, d_u))$ 
22:  $c_2 := \text{Encode}_{d_v}(\text{Compress}_q(\mathbf{v}, d_v))$ 
23: return  $c = (c_1 || c_2)$ 
```

Le KEM prioritaire du NIST

Crystals-Kyber : <https://pq-crystals.org/kyber/>

Structure algébrique : corps cyclotomiques

Recalcul de A via la seed du **PRNG**
(backdoorable ?)

CBD : échantillonneur binomial pour le bruit
Doit être temps constant et masqué

Multiplications via NTT (analogue arithmétique de la FFT), complexité quasi-linéaire
Doit être temps constant et masqué

Algorithm 5 KYBER.CPAPKE.Enc(pk, m, r): encryption

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8 + 32}$

Input: Message $m \in \mathcal{B}^{32}$

Input: Random coins $r \in \mathcal{B}^{32}$

Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$

```
1:  $N := 0$ 
2:  $\hat{\mathbf{t}} := \text{Decode}_{12}(pk)$ 
3:  $\rho := pk + 12 \cdot k \cdot n/8$ 
4: for  $i$  from 0 to  $k - 1$  do
5:   for  $j$  from 0 to  $k - 1$  do
6:      $\hat{\mathbf{A}}^T[i][j] := \text{Parse}(\text{XOF}(\rho, i, j))$ 
7:   end for
8: end for
9: for  $i$  from 0 to  $k - 1$  do
10:   $\mathbf{r}[i] := \text{CBD}_{\eta_1}(\text{PRF}(r, N))$ 
11:   $N := N + 1$ 
12: end for
13: for  $i$  from 0 to  $k - 1$  do
14:   $\mathbf{e}_1[i] := \text{CBD}_{\eta_2}(\text{PRF}(r, N))$ 
15:   $N := N + 1$ 
16: end for
17:  $\mathbf{e}_2 := \text{CBD}_{\eta_2}(\text{PRF}(r, N))$ 
18:  $\hat{\mathbf{r}} := \text{NTT}(\mathbf{r})$ 
19:  $\mathbf{u} := \text{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$ 
20:  $\mathbf{v} := \text{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_2 + \text{Decompress}_q(\text{Decode}_1(m), 1)$ 
21:  $c_1 := \text{Encode}_{d_u}(\text{Compress}_q(\mathbf{u}, d_u))$ 
22:  $c_2 := \text{Encode}_{d_v}(\text{Compress}_q(\mathbf{v}, d_v))$ 
23: return  $c = (c_1 || c_2)$ 
```

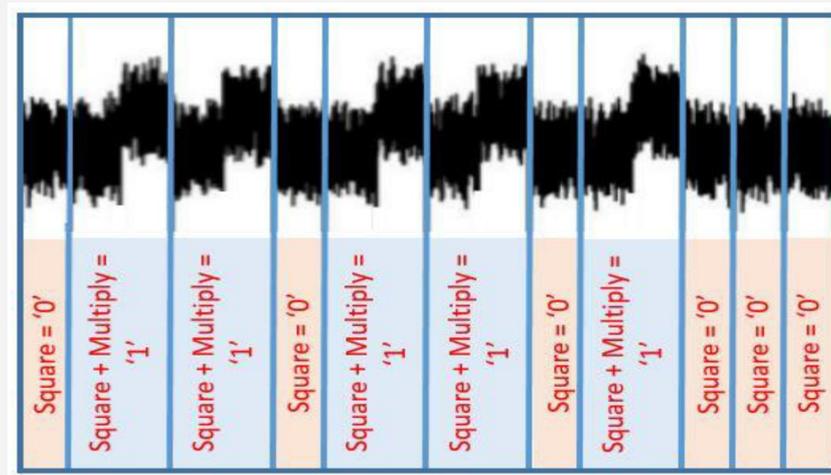
Temps constant ?

Signification : le flot d'instruction et d'accès mémoire doit être **statistiquement indépendant** des données sensibles

- L'exemple typique à ne pas faire :
if (secret bit = 1) do BLA else do BLA

- En contrôlant la consommation énergétique, on peut lire les bits de clés sur les traces

(Image empruntée à *Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman*, M. Randolph & W. Diehl, Cryptography, MDPI, 2020)



- Il faut protéger tout branchement conditionnel ou accès à des zones mémoires sensibles

=> Implémentations plus lentes, plus complexes (trop ?)

Masquer ?

Pour simplifier, masquer = “dupliquer de manière randomisée” des portions de codes.

- Masquage booléen : XOR avec des bits aléatoires, dummy instructions, etc...

- Masquage arithmétique :
$$X = X_1 + \dots + X_{t-1} + X_t \in \mathbb{Z}/B\mathbb{Z}$$
$$\begin{array}{ccc} & \uparrow \$ & \uparrow \$ \\ & \mathbb{Z}/B\mathbb{Z} & \mathbb{Z}/B\mathbb{Z} \end{array}$$

- Souvent nécessaire : conversion booléen \leftrightarrow arithmétique via look-up table (très coûteux !)

Conséquences : implémentations t fois plus grosses, opérations plus lentes (trop ?)

Composants de Kyber et SCA

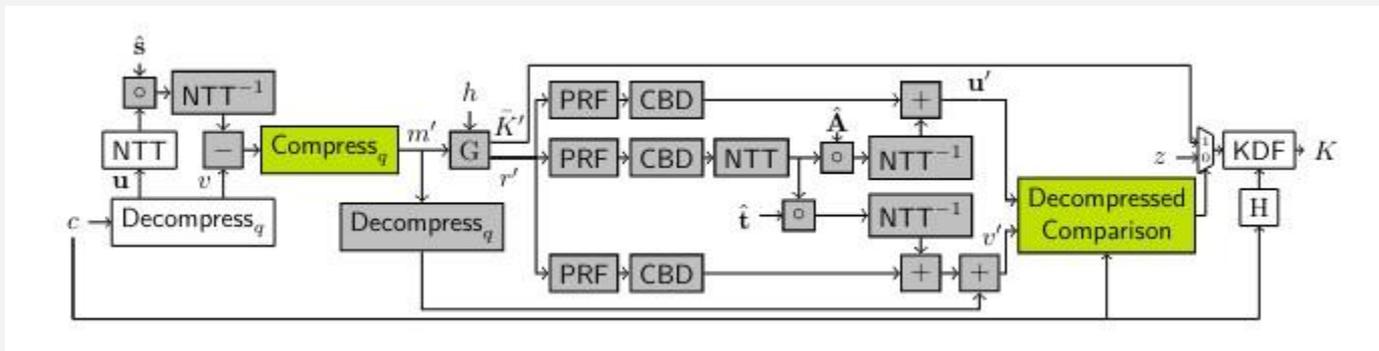


Image empruntée à <https://eprint.iacr.org/2021/483.pdf> : *Masking Kyber, first and higher order implementation*, Bos et al.

Les boîtes en gris et en vert doivent être masquées et implémentées en temps constant

- Régulièrement depuis 2017, attaques side-channel réussies (DPA, par timing, par faute, ...) sur de plus en plus de scénarios (low-end, software, bientôt hardware?)
- D'autre part, ceci est juste le PKE.
 - Kyber est un KEM (échange de clé) obtenu depuis le PKE par une “transformation générique” Elle donne une sécurité **Ind-CCA** au schéma, mais c'est un casse-tête à protéger contre les adversaire side-channel

Challenges pour le déploiement de Kyber

La crypto actuelle et son agencement dans un système sécurisé est extrêmement optimisée et codifiée. Le design formel de Kyber est relativement stable, mais il faut maintenant l'intégrer.

Défi général : comment effectuer la migration crypto à grande échelle sur tous les scénarios ?

Scénarios à “ressources illimitées” :

- Au sein d'un protocole de grande échelle (TLS, ...) : choisir les primitives (hash, PRNG, ...), agencer la crypto dans le protocole, documenter les normes (RFC, IETF, NIST...)
- protocoles hybrides : couche classique + couche post-quantique.
Design pas clarifié, puis prouver/vérifier la sécurité, puis le faire efficacement.

Scénarios à faibles ressources (chipcard, embarqué, ...) :

- choix de l'arithmétique, développement coprocesseurs
- minimiser la bande passante tout en masquant le composant
- tout changement à la hausse => des tonnes de \$\$ perdus pour l'industrie, donc les clients

Un problème complexe : “protéger l'étape de FO transform” (voir <https://eprint.iacr.org/2022/036.pdf>)

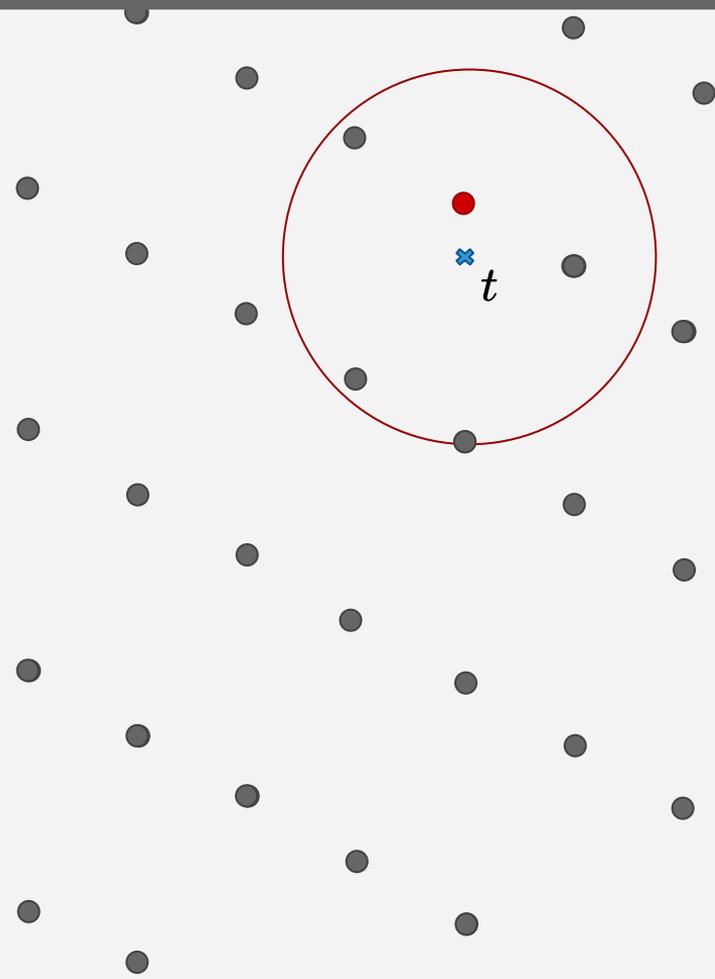
Réseaux euclidiens, partie 2

Closest Vector Problem (CVP)

CVP : trouver $v \in \mathcal{L}$ le plus proche d'une cible $t \in \mathbb{R}^n$

CVP $_{\gamma}$: Pour $\gamma \geq 1$, trouver $v \in \mathcal{L}$ tel que $\|v - t\| \leq \gamma d(t, \mathcal{L})$

On a une réduction : **SVP**_{poly(n)} \longrightarrow **CVP**_{poly(n)} λ_1



Closest Vector Problem (CVP)

CVP : trouver $v \in \mathcal{L}$ le plus proche d'une cible $t \in \mathbb{R}^n$

CVP $_{\gamma}$: Pour $\gamma \geq 1$, trouver $v \in \mathcal{L}$ tel que $\|v - t\| \leq \gamma d(t, \mathcal{L})$

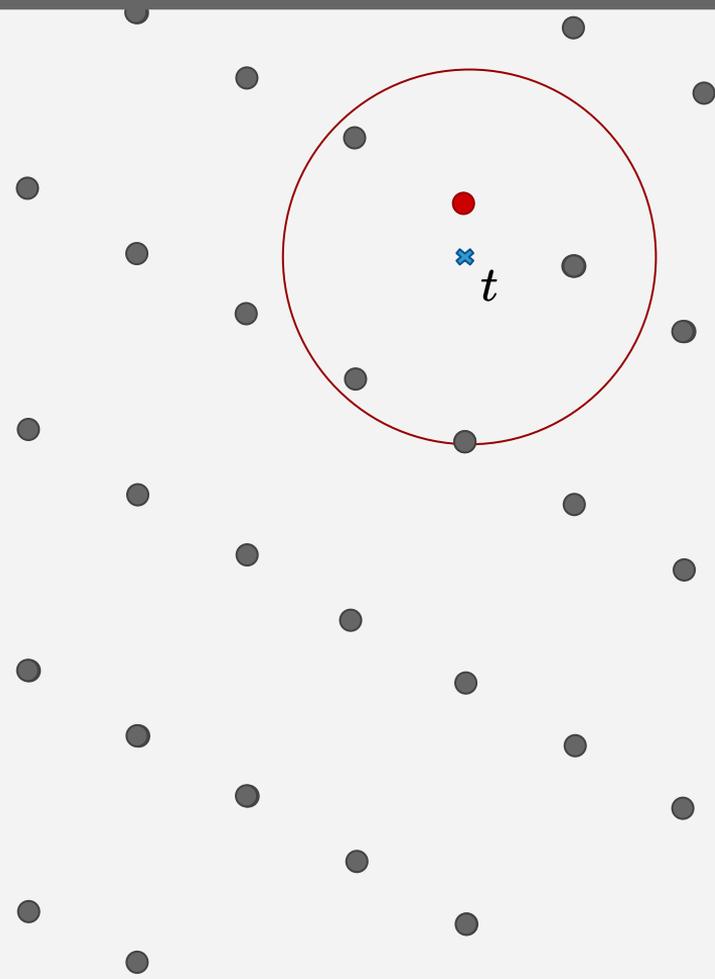
On a une réduction : **SVP**_{poly(n)} \longrightarrow **CVP**_{poly(n)} λ_1

Comment trouver un tel vecteur ? **Problème de décodage**

qualité du décodage \sim plus long vecteur de la base

Si la base est "bien réduite", on arrive à décoder proche.

Complexité \sim coût pour bien réduire \sim infaisable en grande dim

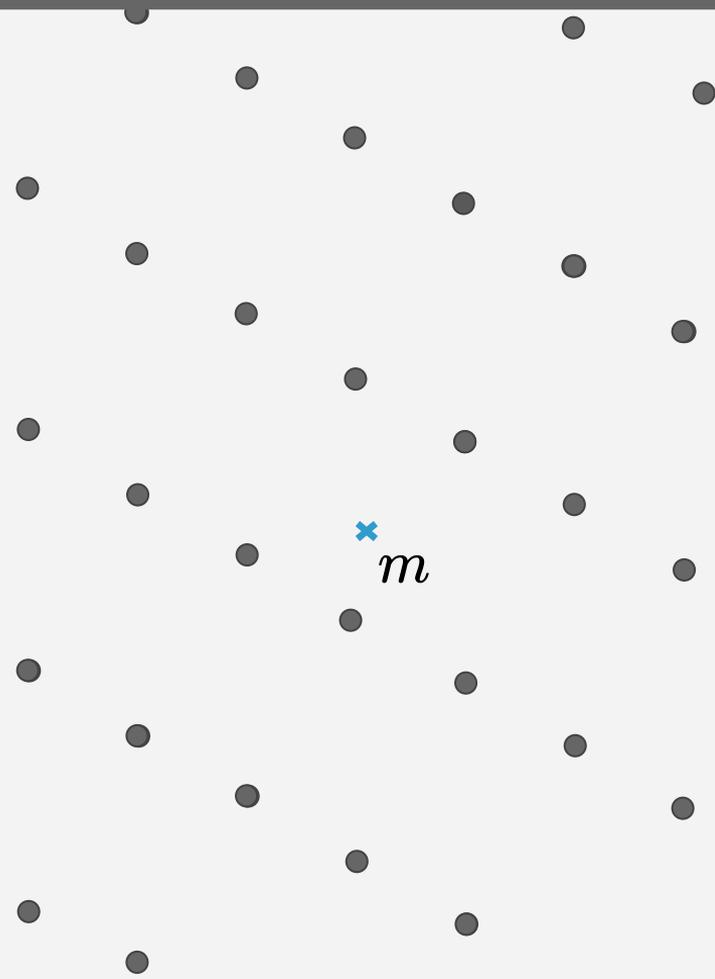


Signatures “Hash-then-sign” avec des réseaux euclidiens

Hash-then-sign

Pour signer un message M

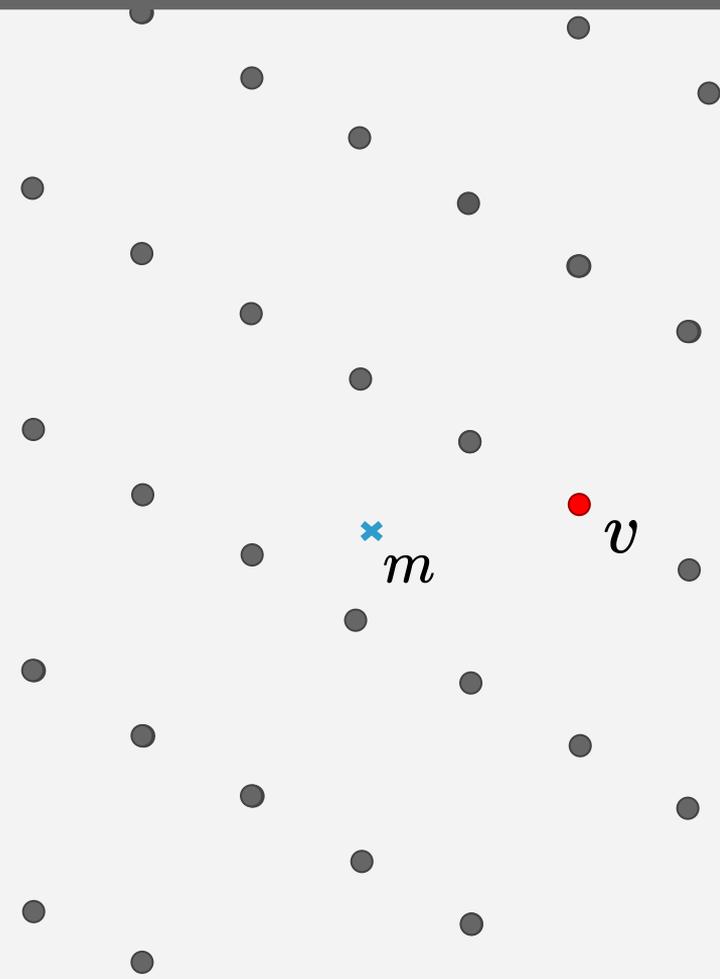
1) "Hasher" le message M dans l'espace ambiant : $m = \mathcal{H}(M)$



Hash-then-sign

Pour signer un message M :

- 1) “Hasher” le message M dans l’espace ambiant : $m = \mathcal{H}(M)$
- 2) Trouver un point v d’un réseau **public** fixé, **proche** de m
- 3) La signature est le vecteur plutôt court $s = m - v$



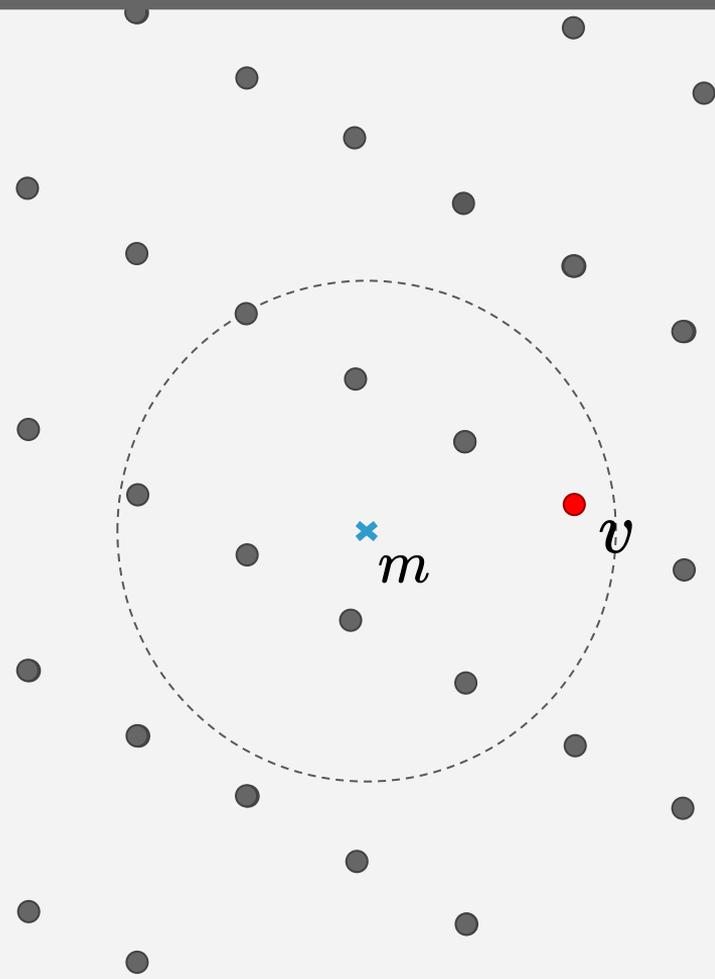
Hash-then-sign

Pour signer un message M :

- 1) “Hasher” le message M dans l’espace ambiant : $m = \mathcal{H}(M)$
- 2) Trouver un point v d’un réseau **public** fixé, **proche** de m
- 3) La signature est le vecteur plutôt court $s = m - v$

Pour vérifier une paire (M, s) :

- 1) Calculer $m = \mathcal{H}(M)$
- 2) Si $m - s$ n’est pas dans le réseau, rejeter.
- 3) Si $\|m - s\|$ est trop grande, rejeter.



Premières difficultés

“Trouver un vecteur du réseau proche d’une cible arbitraire dans \mathbb{R}^n ” ? (C’est un **CVP** $_\gamma$)

Exemple : arrondir les coordonnées (“Round-off”) :

- Si b_1, \dots, b_n est une base du réseau \mathcal{L} , on écrit $m = \sum_i m_i b_i, m_i \in \mathbb{R}$
- Alors $v = \sum_i \lfloor m_i \rfloor b_i \in \mathcal{L}$, et on a $\|m - v\| \leq \frac{n}{2} \max \|b_i\|$.

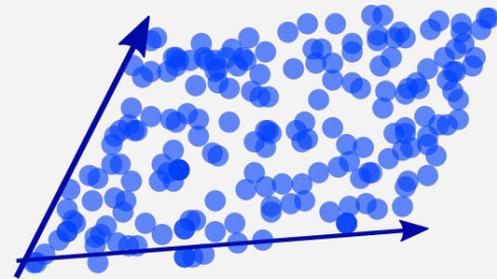
=> Il faut une base **B** secrète de vecteurs courts, et une base publique “mauvaise” **A** pour vérifier.

(la base secrète est aussi appelée *une trappe*)

Signatures déterministes et (in)sécurité

Si on signe avec un algorithme comme Round-off :

- 1) *Chaque signature donne de l'information sur la base secrète !*
- 2) Avec assez de signatures, on commence à “voir” son domaine fondamental.
- 3) Par apprentissage statistique, on récupère⁽¹⁾ cette base => **Cassé !**



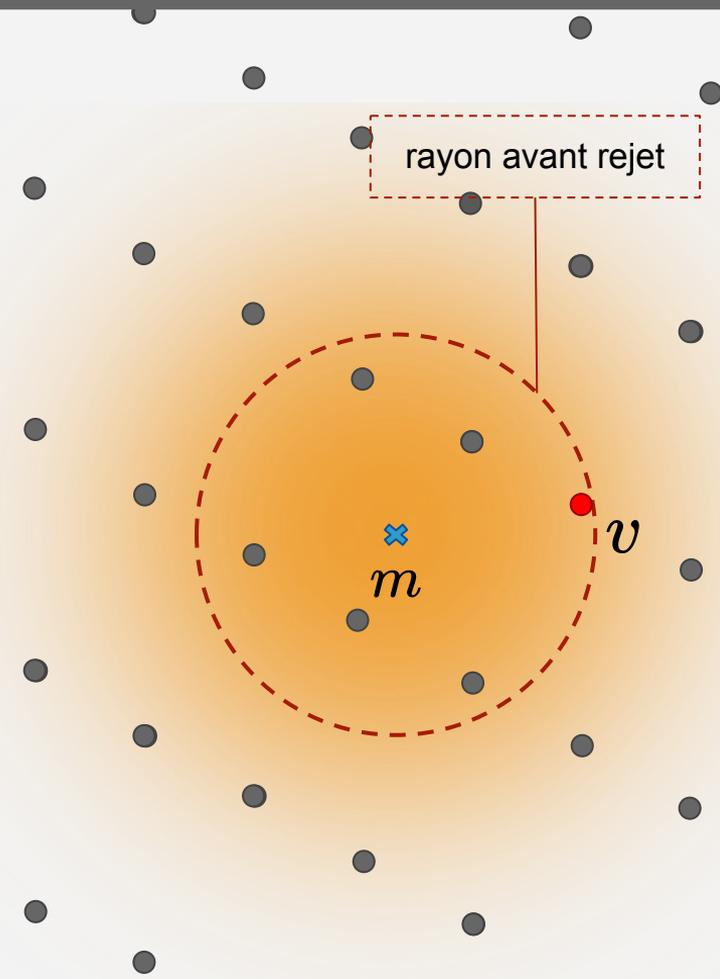
=> Signer sans déterminisme et sans faire fuiter la base secrète

(1) NGuyen-Regev, 2006, Ducas-NGuyen, 2012, d'autres articles récents

La solution de Gentry, Peikert et Vaikuntanathan (2008)

Ingrédient 1 : Signer = échantillonner des points aléatoires avec une distribution

- **supportée sur le réseau public**
- **Gaussienne discrète sphérique**
- **centrée autour du hashé**



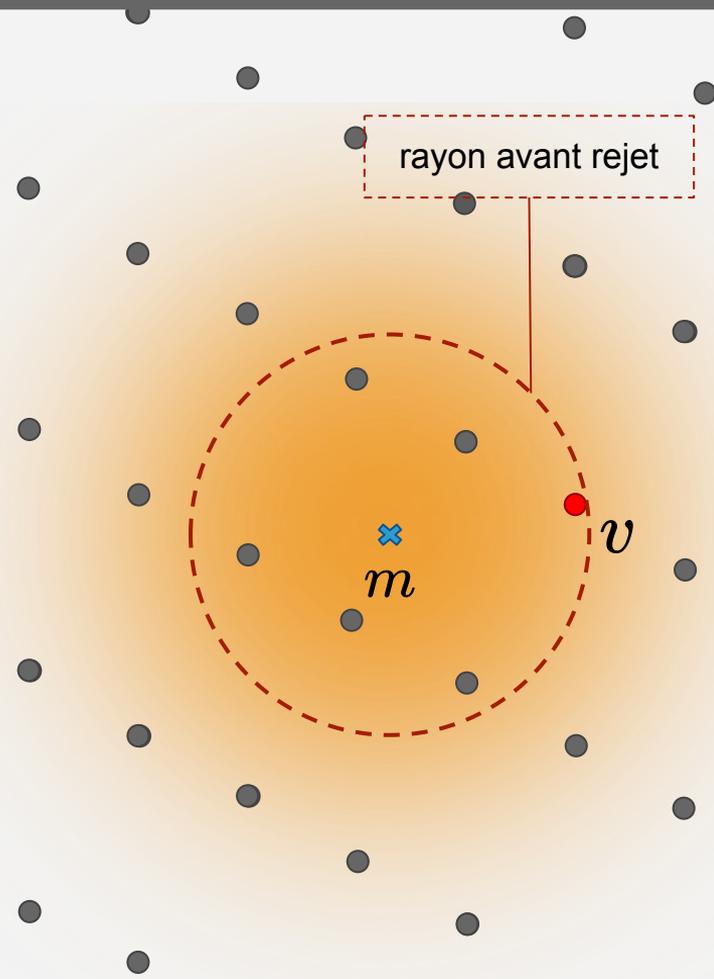
La solution de Gentry, Peikert et Vaikuntanathan (2008)

Ingrédient 1 : Signer = échantillonner des points aléatoires avec une distribution

- **supportée sur le réseau public**
- **Gaussienne discrète sphérique**
- **centrée autour du hashé**

Ingrédient 2 : **utiliser des réseaux SIS**

- Si \mathbf{B} est une base d'un réseau SIS, $\mathbf{A}\mathbf{B} = 0 \pmod q$
 $\mathbf{A}s = \mathbf{A}m \pmod q \Rightarrow$ suffit de signer **une pré-image** de m
- Permet **une preuve de sécurité théorique** :
"Forger \Rightarrow résoudre une instance aléatoire de SIS
 \Rightarrow calculer des vecteurs courts dans n'importe quel réseau"



De GPV au futur standard

Limites de GPV :

1. Toujours l'algèbre linéaire
2. KeyGen = générer des paires (\mathbf{A}, \mathbf{B}) avec \mathbf{B} courte : c'est compliqué !
3. on va vraiment générer des Gaussiens en dimension 1000 ??
4. Selon l'échantillonneur, tailles des données absurdes

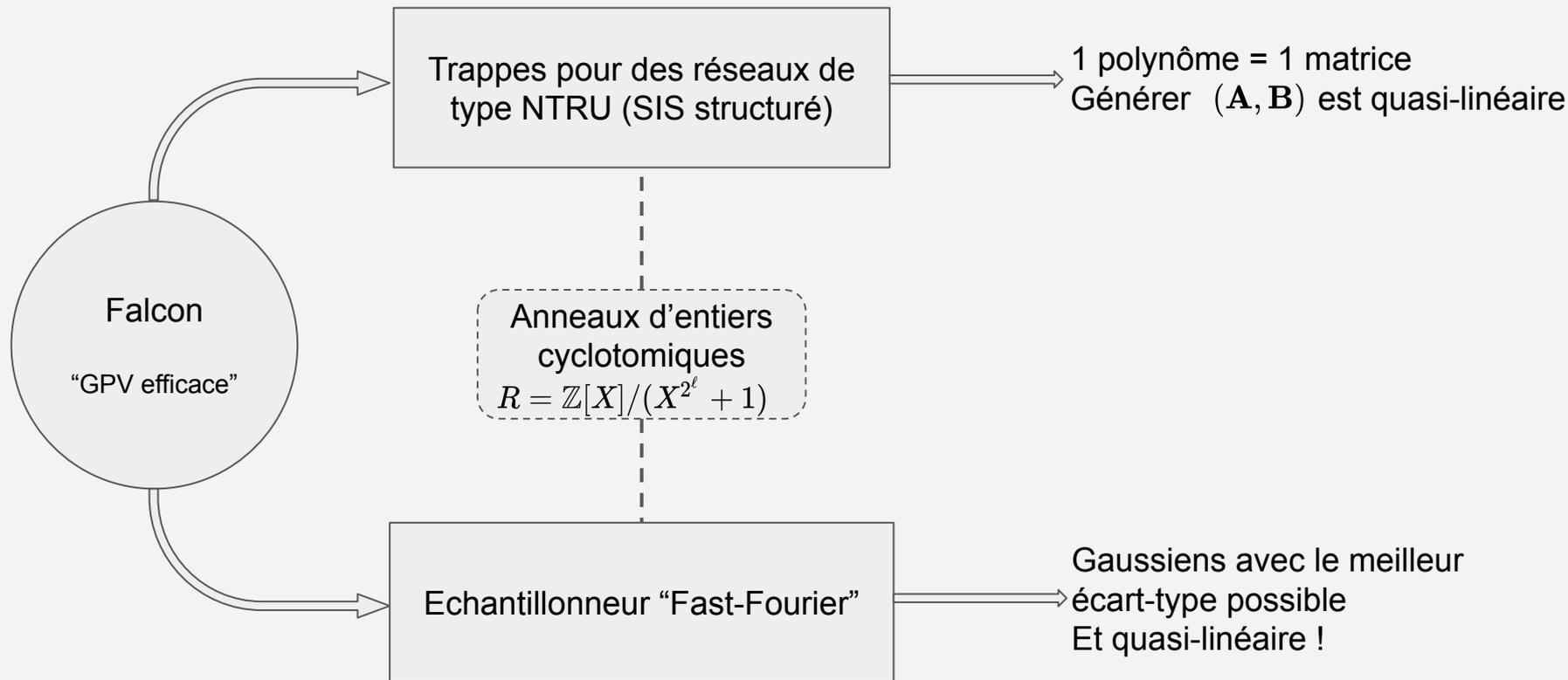
Solutions:

pour 1. et 4. : utiliser des réseaux algébriquement structurés (et des techniques supplémentaires)

pour 2. et 3.: structures + des maths + de la sueur, mais le sujet est plutôt bien compris.

Falcon

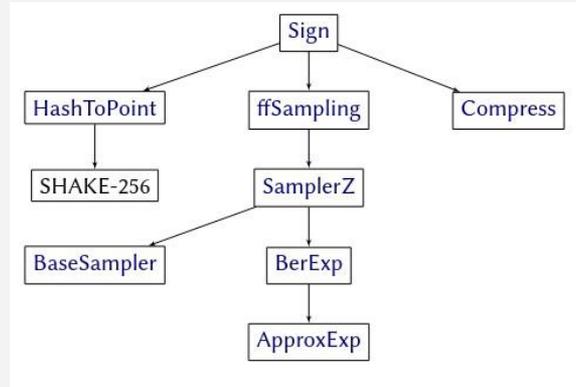
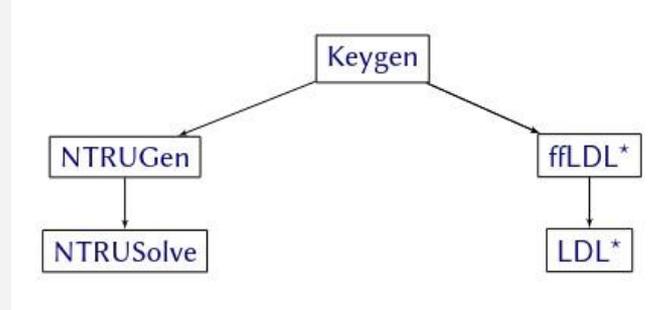
“Falcon: a quest for compactness”



Falcon de manière synthétique

Falcon : <https://falcon-sign.info/>

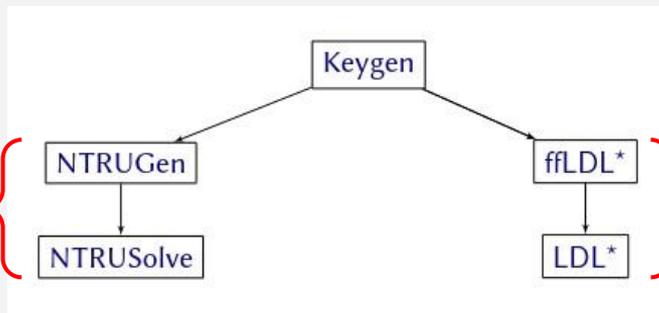
Pas le standard prioritaire, mais sera standardisé ensuite.



Falcon de manière synthétique

Falcon : <https://falcon-sign.info/>

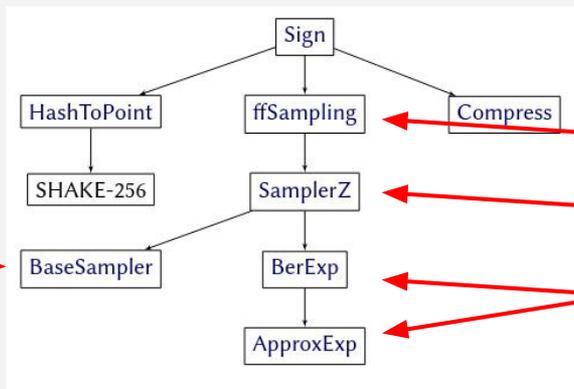
cher, compliqué,
utilise des floats



données de l'échantillonneur
sous forme d'arbre de floats

NB : floats & constant-time = enfer

samplieur précalculé
dans une table



sensible, compliqué, utilise l'arbre

sensible, fait du rejet, manipule
des écart-types secrets

samplieur bernoulli par
approximation polynomiale

“Pourquoi standardiser ça ???”

Falcon

compliqué, utilise des floats,
masquage prohibitif

Signatures compactes
Meilleure signature+clé du NIST
Vérification plus rapide



Crystals-Dillithium

Autre paradigme de signature avec SIS
Plus simple, plus flexible
rapide, pas de floats

Signatures de plusieurs KBytes
Clés de plusieurs KBytes
Echantillonnage par rejet lent si masqué

“Pourquoi standardiser ça ???”

Falcon

compliqué, utilise des floats,
masquage prohibitif

Signatures compactes
Meilleure signature+clé du NIST
Vérification plus rapide



Crystals-Dillithium

Autre paradigme de signature avec SIS
Plus simple, plus flexible
rapide, pas de floats

Signatures de plusieurs KBytes
Clés de plusieurs KBytes
Echantillonnage par rejet lent si masqué

Les signatures, signatures+clés de Dillithium sont trop grosses pour de nombreux scénarios.
(Comparer à 32 Bytes < ECDSA, RSA < 128 Bytes)

Le besoin de floats de Falcon limite les cas d'usage. Les signatures font toujours ~600 Bytes.

Problème : on n'a rien de mieux en post-quantique !

Récents progrès sur ces design

Falcon



Mitaka (2021)

masquage moins cher
implem simple, moins rigide
possible sans floats (?)
même taille de signature/clés

Crystals-Dillithium

Récents progrès sur ces design

Falcon



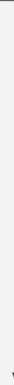
Mitaka (2021)



Solmae

- Meilleure sécurité
- Signatures encore plus courtes
- Clés plus courtes aussi
- Toujours possible sans floats
- Masquage toujours délicat

Crystals-Dillithium



Haetae

- Rejet moins cher
- Signatures plus courtes
- Toujours simple et rapide
- clés, signatures toujours grosses

Compétition KPQC



“Il est temps de conclure...”

Etat de la PQC : résumé

Principaux problèmes/défis :

- un ordre de grandeur perdu en bande passante avec les réseaux euclidiens
- autres paradigmes moins stables, et pour le moment moins performants
- robustesse side-channel : sujet complexe et très jeune

vs

la migration doit commencer le plus rapidement possible

“Les experts attendent des calculs quantiques probants sous 15 ans !”

Compétition NIST :

- Kyber (KEM), Dilithium (signature) **en cours de standardisation**
- Ensuite : Falcon (signature), SPHINCS (signature)
- (Puis les “réservistes” ?)

Challenges
d’implémentation, de
protection, de
déploiement

Autres compétitions en cours :

- 4e tour NIST pour signatures : tout sauf réseaux euclidiens
- Compétitions d’autres standards

Challenges de
conception et
d’implémentation

Le moment publicitaire

Plusieurs aspects : travail de (re)conception, travail sur la migration, analyse/attaque/contre-mesures.

- Attaques ? matériel sérieux pour monter une attaque side-channel, acquérir les traces, etc.
- Peut se combiner aux algorithmes de cryptanalyse
- Depuis quelques années : on utilise du ML pour raffiner, améliorer l'efficacité.

Qui s'en occupe en France ? (et où faire des stages, des thèses...)

- Organismes étatiques : ANSSI (demander à Guénaël), DGA, ...
- CESTI : CEA, Quarkslab, ...
- Entreprises de cryptographie et cybersécurité :
 - les grands groupes : Orange, Thalès, Idemia, ...
 - startups : PQShield, ...
- Laboratoires académiques : IRISA (Rennes, me demander), DIX (les collègues ici), IMB (Bordeaux), ENS Lyon, ...

Pour finir

