Le calcul sécurisé, troisième cours

Chiffrement totalement homomorphe: calculer sur des données chiffrées (2e partie)

Xavier Leroy

2025-11-20

Collège de France, chaire de sciences du logiciel
xavier.leroy@college-de-france.fr

Rappels du cours précédent

L'atelier de bijouterie d'Alice

(C. Gentry, CACM 53(3), 2010)

(source)

Alice la bijoutière place les matériaux précieux dans une boîte à gants qu'elle verrouille avec sa clé privée.

Bob l'ouvrier peut assembler le bijou mais pas partir avec.

Lorsque le bijou est terminé, Alice déverrouille la boîte et le récupère.

L'atelier de bijouterie d'Alice

(C. Gentry, CACM 53(3), 2010)

(source)

Problème : les gants durcissent rapidement lorsqu'on s'en sert.

Ils deviennent complètement rigides et inutilisables avant que le bijou ne soit terminé.

L'atelier de bijouterie d'Alice

(C. Gentry, CACM 53(3), 2010)

(source)

Solution : Alice place la boîte dans une deuxième boîte, avec la clé de la première boîte, et verrouille la deuxième boîte.

Bob déverrouille la première boîte, en extrait le bijou et les matériaux, et continue son travail.

On répète cette procédure jusqu'à ce que le bijou soit terminé.

L'atelier de bijouterie d'Alice

(C. Gentry, CACM 53(3), 2010)

(source)

Problème : les cadenas d'Alice sont compliqués et rouillés. Bob n'arrive pas à les ouvrir avant que ses gants durcissent.

Solution : Alice met, parmi les outils, une burette d'huile pour graisser le cadenas et faciliter l'ouverture.

Attention à ce que l'huile ne diminue pas la sécurité des boîtes !

Rappel : Construction d'un chiffrement basé sur le bruit

Le chiffré d'un bit b avec la clé p est

$$\mathcal{E}_p(b) = pq + 2r + b \qquad \begin{array}{l} q \text{ entier naturel aléatoire} \gg p \\ r \text{ entier relatif aléatoire, } |r| < p/4 \end{array}$$

De la forme $\textit{chiffré} = \textit{bruit}_1 \cdot \textit{clé} + \textit{bruit}_2$ où le message clair b est inclus dans $\textit{bruit}_2$ et masqué par $\textit{bruit}_1$.

On peut déchiffrer tant que $|r| < p/4$, en annulant $\textit{bruit}_1$:

$$\mathcal{D}_p(c) = (c - p\lfloor c/p \rceil) \bmod 2$$

Ce chiffrement est faiblement homomorphe

$$\mathcal{E}_p(b_1) + \mathcal{E}_p(b_2) = \text{un chiffré de } b_1 \oplus b_2$$
$$\mathcal{E}_p(b_1) \cdot \mathcal{E}_p(b_2) = \text{un chiffré de } b_1 \cdot b_2$$

tant que le bruit r des résultats reste $< p/4$.

Le bruit $N(c)$ (en bits) augmente

- peu lors d'une addition $\max(N(c_1), N(c_2)) + 1$
- beaucoup lors d'une multiplication $N(c_1) + N(c_2) + 1$

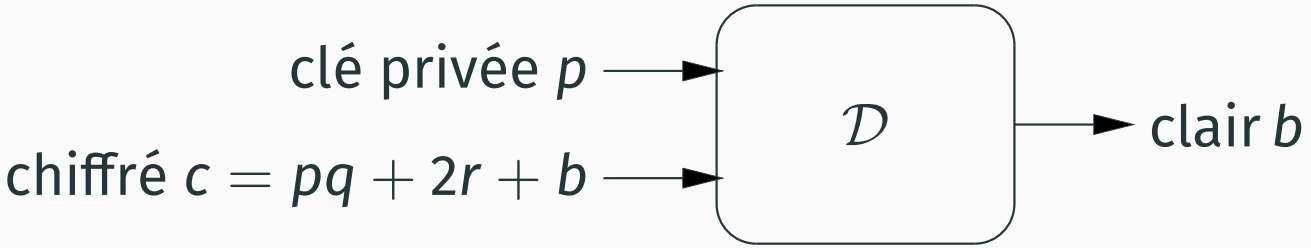
Que peut-on évaluer homomorphiquement avec ce chiffrement ?

- Des polynômes multivariés de degré faible ($\leq \lambda$);
  peu de limites sur le nombre de monômes.

- Des circuits booléens de faible profondeur multiplicative (AND, OR) (entre $\log_2 \lambda$ et λ suivant la forme du circuit);
  peu de limites sur la profondeur additive (XOR, NOT).

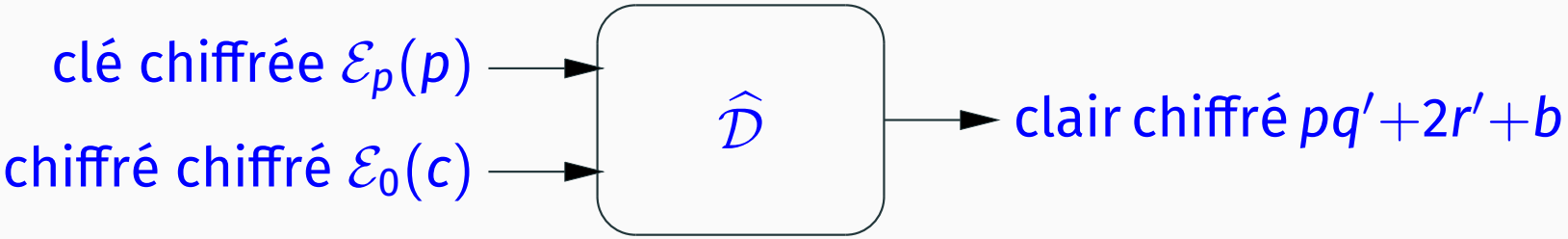
Rappel : La procédure de bootstrap (Gentry, 2009)

Réduire le bruit d'un chiffré par évaluation homomorphe du circuit de déchiffrement $\mathcal{D}$.



Rappel : La procédure de bootstrap (Gentry, 2009)

Réduire le bruit d'un chiffré par évaluation homomorphe du circuit de déchiffrement $\mathcal{D}$.



Le résultat est un chiffré équivalent à c, mais dont le bruit dépend uniquement de la profondeur multiplicative du circuit $\widehat{\mathcal{D}}$.

(Note : $\mathcal{E}_0$ est le chiffrement trivial $\mathcal{E}_0(b) = b$, approprié si b ne révèle pas d'information.)

Chiffrement totalement homomorphe ?

Chiffrement faiblement homomorphe + bootstrap

$\Rightarrow$ chiffrement totalement homomorphe

(qui peut évaluer des circuits booléens arbitraires)

À condition que le circuit de déchiffrement $\mathcal{D}$ soit de profondeur multiplicative suffisamment faible !

Proposition de Gentry : «graisser le chiffrement»
(ajouter aux chiffrés des informations redondantes)
pour simplifier le circuit de déchiffrement.

$$\mathcal{D}_p(c, z_1 \dots z_N) = LSB(\lfloor \sum_i s_i \cdot z_i \rceil) \oplus LSB(c)$$

Ce cours : Améliorer l'approche

1. De meilleurs chiffrements faiblement homomorphes

À base du problème LWE (*Learning With Errors*).

2. Des multiplications qui augmentent moins le bruit

Approche BGV, avec un chiffrement étagé (*leveled*).

3. Une procédure de bootstrap plus efficace

Approche TFHE, avec «bootstrap programmable»
(réduction du bruit tout en évaluant des fonctions tabulées).

Le problème LWE
(*Learning With Errors*)

Le problème LWE (Oded Regev, 2005)

Résoudre un système d'équations linéaires «approchées» dans les entiers modulo q.

Exemple :

$$
\begin{aligned}
14s_1 + 15s_2 + 5s_3 + 2s_4 &\approx 8 \pmod{17} \\
13s_1 + 14s_2 + 14s_3 + 6s_4 &\approx 16 \pmod{17} \\
6s_1 + 10s_2 + 13s_3 + 1s_4 &\approx 3 \pmod{17} \\
10s_1 + 4s_2 + 12s_3 + 16s_4 &\approx 12 \pmod{17} \\
9s_1 + 5s_2 + 9s_3 + 6s_4 &\approx 9 \pmod{17} \\
3s_1 + 6s_2 + 4s_3 + 5s_4 &\approx 16 \pmod{17} \\
6s_1 + 7s_2 + 16s_3 + 2s_4 &\approx 3 \pmod{17}
\end{aligned}
$$

où $\approx$ signifie «égal à 1 près». (Solution : $\mathbf{s} = (0, 13, 9, 11)$.)

Le problème LWE

$$14s_1 + 15s_2 + 5s_3 + 2s_4 = 8 \pmod{17} \quad (1)$$

$$13s_1 + 14s_2 + 14s_3 + 6s_4 = 16 \pmod{17} \quad (2)$$

$$6s_1 + 10s_2 + 13s_3 + 1s_4 = 3 \pmod{17} \quad (3)$$

$$10s_1 + 4s_2 + 12s_3 + 16s_4 = 12 \pmod{17} \quad (4)$$

Si l'erreur est égale à zéro, le problème se résout facilement par élimination gaussienne, dès que l'on a 4 équations indépendantes.

Le problème LWE

$$14s_1 + 15s_2 + 5s_3 + 2s_4 = 8 \pm 1 \pmod{17} \tag{1}$$

$$13s_1 + 14s_2 + 14s_3 + 6s_4 = 16 \pm 1 \pmod{17} \tag{2}$$

$$6s_1 + 10s_2 + 13s_3 + 1s_4 = 3 \pm 1 \pmod{17} \tag{3}$$

$$10s_1 + 4s_2 + 12s_3 + 16s_4 = 12 \pm 1 \pmod{17} \tag{4}$$

Si l'erreur est non nulle, elle explose pendant l'élimination gaussienne. Par exemple, $3 \times (4) - 5 \times (3)$ élimine s_1 et donne

$$13s_2 + 5s_3 + 9s_4 = 4 \pm 8 \pmod{17}$$

Toutes les valeurs du membre gauche sont possibles (modulo 17).

Le problème LPN (*Learning Parity with Noise*)

LWE généralise un problème plus ancien : LPN.

Une fonction de parité est un XOR de certaines de ses entrées :

$$P_{3,4,7}(x_1, \ldots, x_8) = x_3 \oplus x_4 \oplus x_7$$

C'est un produit scalaire avec un vecteur $\mathbf{s}$ de bits, qui caractérise la fonction :

$$P_{\mathbf{s}}(x_1, \ldots, x_n) = \sum_{i=1}^{n} s_i \cdot x_i \pmod 2$$

Le problème LPN (*Learning Parity with Noise*)

Le problème LP (*Learning Parity*) : étant donnés des échantillons $(\mathbf{x}, P_{\mathbf{s}}(\mathbf{x}))$ pour $\mathbf{x}$ choisi aléatoirement, retrouver $\mathbf{s}$.

Facile par élimination gaussienne si on a n échantillons indépendants. Difficile pour les méthodes d'apprentissage par descente de gradient.

Le problème LPN (*Learning Parity with Noise*)

Le problème LP (*Learning Parity*) : étant donnés des échantillons $(\mathbf{x}, P_{\mathbf{s}}(\mathbf{x}))$ pour $\mathbf{x}$ choisi aléatoirement, retrouver $\mathbf{s}$.

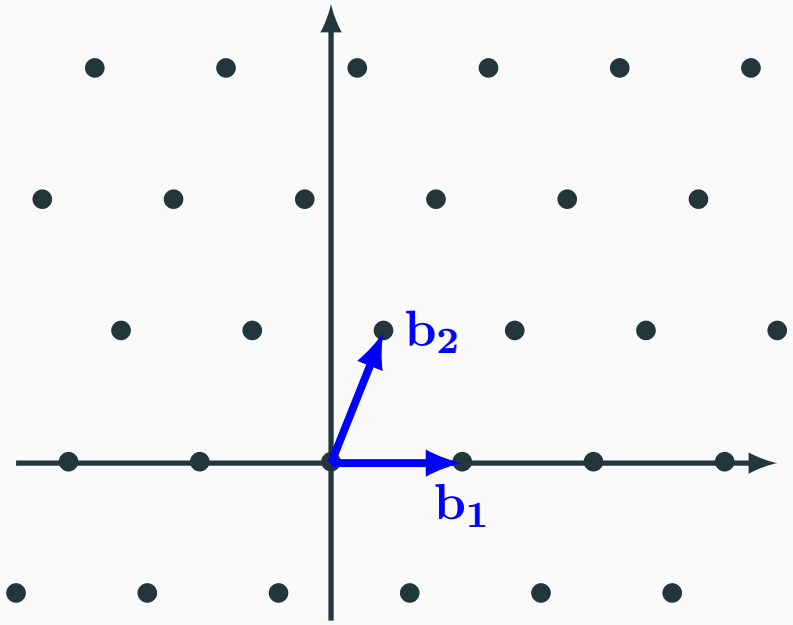
Le problème LPN (*Learning Parity with Noise*) : retrouver $\mathbf{s}$ étant donnés des échantillons «bruités»

$$(\mathbf{x}, P_{\mathbf{s}}(\mathbf{x}) + e) \qquad \begin{array}{l} \mathbf{x} \text{ aléatoire} \\ e = 1 \text{ avec probabilité } \varepsilon, e = 0 \text{ sinon} \end{array}$$

Problème réputé difficile, même dans le cas moyen, même avec un ordinateur quantique.

Les meilleurs algorithmes connus sont en temps et espace sous-exponentiels.

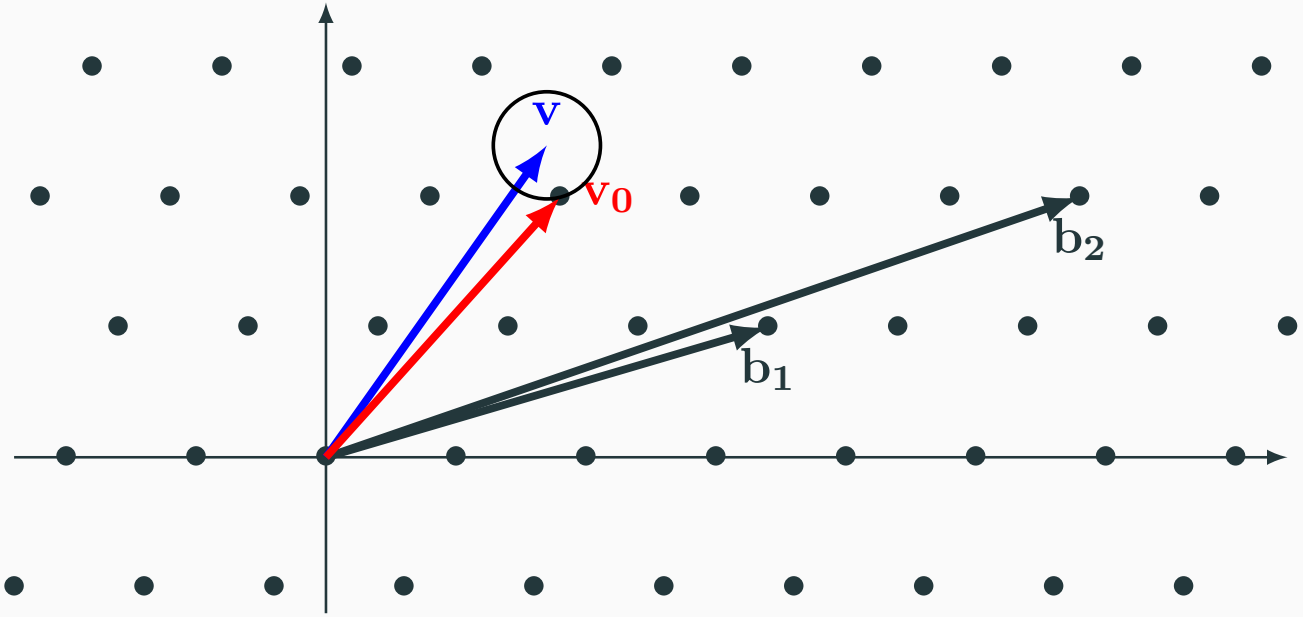
Les réseaux euclidiens



Un réseau euclidien (*lattice*) = l'ensemble des vecteurs à coordonnées entières dans une base $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$ donnée.

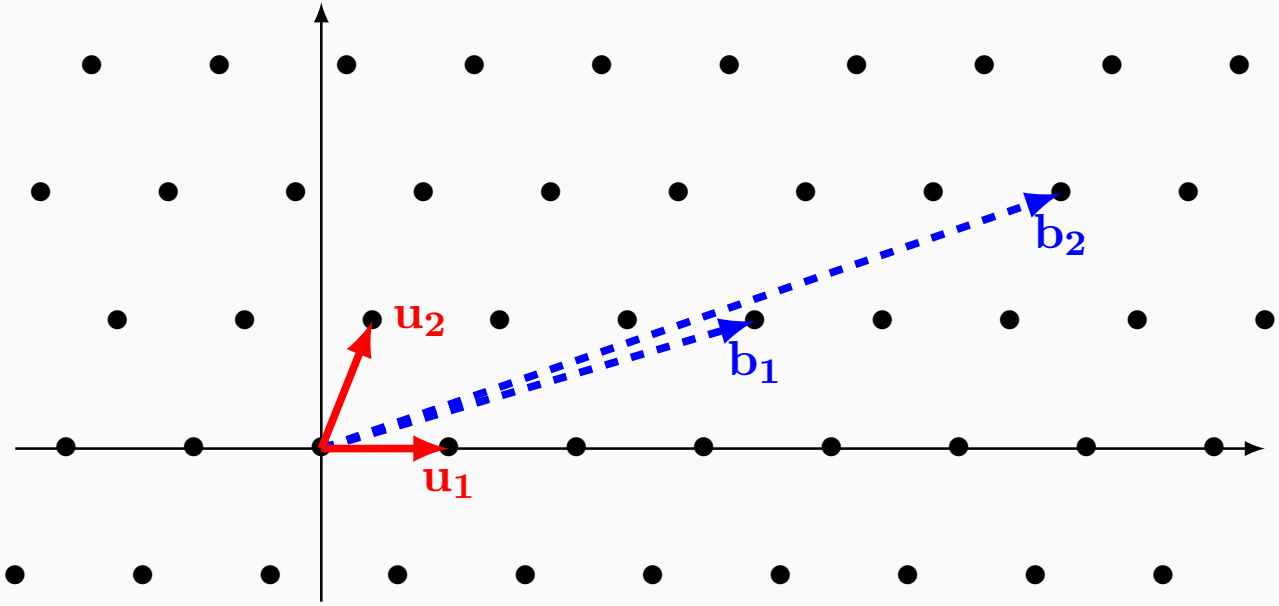
$$\left\{ \sum_{i=1}^{n} p_i \, \mathbf{b}_i \;\middle|\; p_i \in \mathbb{Z} \right\}$$

Le problème CVP (*Closest Vector Problem*)



Étant donné un vecteur **v**, trouver les coordonnées d'un vecteur $\mathbf{v}_0$ appartenant au réseau et le plus proche de **v**.

Le problème SIVP (*Shortest Independent Vectors Problem*)



Trouver *n* vecteurs indépendants de longueur minimale.

Autres problèmes difficiles sur les réseaux euclidiens

SVP (*Shortest Vector Problem*) : trouver le plus petit vecteur non nul du réseau. (Ou juste déterminer sa longueur.)

GapSVP : est-ce que la longueur du plus petit vecteur est ≤ 1 ou $\geq \beta$ pour $\beta > 1$ donné ?

Tous ces problèmes sont NP-difficiles, même approchés.

Des réductions difficile dans le cas le pire $\implies$ difficile dans le cas moyen. (Ajtai, 1995).

LWE et CVP

LWE peut être vu comme une généralisation du problème CVP dans $\mathbb{Z}_q$ (les entiers modulo q).

Résoudre

$$a_{11}x_1 + \cdots a_{1n}x_n \approx b_1$$
$$\vdots$$
$$a_{n1}x_1 + \cdots a_{nn}x_n \approx b_n$$

c'est trouver les coordonnées $\mathbf{x}$ du point du réseau engendré par la base $\mathbf{a}_1, \ldots, \mathbf{a}_n$ qui est à faible distance du point $\mathbf{b}$.

Le problème LWE

Soit $\mathbf{s} \in \mathbb{Z}_q^n$ le secret. On considère des nombres de la forme

$$\langle \mathbf{a}, \mathbf{s} \rangle + e \pmod{q}$$

$\mathbf{a} \in \mathbb{Z}_q^n$ vecteur aléatoire uniforme
$e \in \mathbb{Z}$ entier aléatoire de distribution χ

Problème LWE : étant donnés un ensemble de couples $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, retrouver $\mathbf{s}$.

Problème LWE décisionnel : distinguer entre un ensemble de couples $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$ et un ensemble de couples $(\mathbf{a}, u)$ où $u \in \mathbb{Z}_q$ est aléatoire uniforme.

Quelques résultats de difficulté

(O. Regev, *On lattices, learning with errors, random linear codes, and cryptography*, J. ACM, 2006).

- Si q est exponentiel en n, LWE est au moins aussi difficile que GapSVP approché à un facteur polynomial près.
- Si q est polynomial en n, LWE est au moins aussi difficile que GapSVP ou SIVP approchés à un facteur polynomial près sur un ordinateur quantique.
- LWE décisionnel est aussi difficile que LWE.
- LWE décisionnel est aussi difficile dans le cas moyen que dans le cas le pire.

Chiffrements à base de LWE

Chiffrer avec LWE

Idée générale : cacher le message clair dans le terme d'erreur.

Chiffrement symétrique (avec la clé secrète **s**) :

$$\mathcal{E}_{\mathbf{s}}(m) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + f(m))$$

où $\mathbf{a} \in \mathbb{Z}_q^n$ est un vecteur aléatoire
et $f : Msg \to \mathbb{Z}_q$ une fonction randomisée inversible additive.

Déchiffrement :

$$\mathcal{D}_{\mathbf{s}}((\mathbf{a}, b)) = f^{-1}(b - \langle \mathbf{a}, \mathbf{s} \rangle)$$

La sécurité IND-CPA repose sur le problème LWE décisionnel :
pour qui ne connaît pas **s**, $\mathcal{E}_{\mathbf{s}}(m)$ est indistinguable de $(\mathbf{a}, b)$ avec **a** et b uniformes.

Encodages du message dans le terme d'erreur

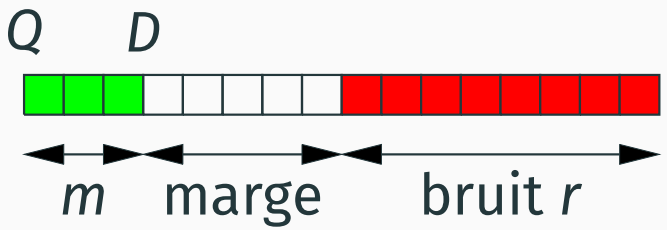
Encodage grand-boutiste (*big endian*) : m dans les poids forts.

$$f(m) = \Delta m + r \quad \text{avec } r \text{ aléatoire, } |r| < \Delta/2$$

$$f^{-1}(c) = \lfloor c/\Delta \rceil$$

Les messages m sont des entiers dans $\mathbb{Z}_t$ avec $t = q/\Delta$.

Exemple si $q = 2^Q$ et $t = 2^T$:
les message m sont des mots de T bits, stockés dans les T bits de poids fort ; le bruit r est un entier signé d'au plus $D = Q - T$ bits.



Encodages du message dans le terme d'erreur

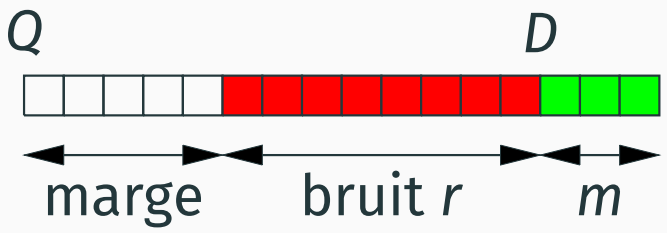
Encodage petit-boutiste (*little endian*) : m dans les poids faibles.

$$f(m) = \Delta r + m \quad \text{avec } r \text{ aléatoire, } |r| < q/2\Delta$$

$$f^{-1}(c) = c \bmod \Delta$$

Les messages m sont des entiers dans $\mathbb{Z}_\Delta$.

Exemple si $q = 2^Q$ et $\Delta = 2^D$:
les message m sont des mots de D bits, stockés dans les D bits de poids faible ; le bruit r est un entier de $Q - D$ bits.



Chiffrement trivial

Si le message m ne révèle pas d'information confidentielle (p.ex. c'est une constante, ou bien il est déjà chiffré), on a des chiffrements triviaux

$$\mathcal{E}_0(m) = (\mathbf{0}, m) \qquad \text{dans le cas petit-boutiste}$$

$$\mathcal{E}_0(m) = (\mathbf{0}, \Delta m) \qquad \text{dans le cas grand-boutiste}$$

qui se déchiffrent en m quelle que soit la clé secrète $\mathbf{s}$.

Chiffrement à clé publique

Une clé publique = k chiffrements de 0.

$$pk = \big((\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s}\rangle + f(0)), \dots (\mathbf{a}_k, \langle \mathbf{a}_k, \mathbf{s}\rangle + f(0))\big)$$

Chiffrer m avec la clé publique $pk = (\mathbf{a}_1, b_1), \dots, (\mathbf{a}_k, b_k)$:
tirer un sous-ensemble $P \subseteq \{1, \dots, k\}$ et prendre

$$\mathcal{E}_{pk}(m) = \big(\sum_{\mathbf{i} \in P} \mathbf{a}_i, \sum_{\mathbf{i} \in P} b_i + m\big) \qquad \text{petit-boutiste}$$

$$\mathcal{E}_{pk}(m) = \big(\sum_{\mathbf{i} \in P} \mathbf{a}_i, \sum_{\mathbf{i} \in P} b_i + \Delta m\big) \qquad \text{grand-boutiste}$$

Chiffrer un vecteur de n messages

On peut chiffrer un vecteur $\mathbf{m}$ de n messages comme on chiffrerait n messages $m_1, \ldots, m_n$:

$$\mathcal{E}_{\mathbf{s}}(\mathbf{m}) = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{f}(\mathbf{m}))$$

où $\mathbf{A}$ est une matrice $n \times n$ aléatoire
et $\mathbf{A} \cdot \mathbf{s}$ l'application de $\mathbf{A}$ au vecteur $\mathbf{s}$.

Problème : le chiffré est de taille quadratique ($n^2 + n$ entiers).

(P.ex. $n = 1024$ et $q = 2^{64} \Rightarrow$ 8396800 octets.)

Chiffrer avec RLWE (*Ring Learning With Error*)

Idée : au lieu de matrices **A** aléatoires quelconques, utilisons des matrices de la forme suivante :

$$\mathbf{A} = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ -a_{n-1} & a_0 & a_1 & \cdots & a_{n-2} \\ -a_{n-2} & -a_{n-1} & a_0 & \cdots & a_{n-3} \\ \vdots & \vdots & \vdots & & \vdots \\ -a_1 & -a_2 & -a_3 & \cdots & a_0 \end{pmatrix}$$

avec $a_0, \ldots, a_{n-1}$ tirés uniformément dans $\mathbb{Z}_q$

Cette matrice ne sort pas d'un chapeau : elle correspond à la multiplication de polynômes modulo $X^n + 1$ (i.e. $X^n = -1$).

Chiffrer avec RLWE (*Ring Learning With Error*)

Le secret S, l'aléa A, le bruit R, le texte clair M et le chiffré C sont tous vus comme des polynômes dans $Z_q[X]/(X^n + 1)$.

$$\mathcal{E}_S(M) = (A, AS + \Delta R + M) \qquad \text{petit-boutiste}$$

$$\mathcal{E}_S(M) = (A, AS + \Delta M + R) \qquad \text{grand-boutiste}$$

Coefficients de A : aléatoires dans $\mathbb{Z}_q$.
Coefficients de R : aléatoires, pas trop grands.

Coefficients de M : dans $\mathbb{Z}_\Delta$ (petit-boutiste) ou $\mathbb{Z}_{q/\Delta}$ (grand-boutiste).

La taille du chiffré est linéaire ($2n$ entiers).

Sécurité prouvée par la difficulté du problème RLWE (problème LWE pour les polynômes) (si n est une puissance de 2).

GLWE (*Generalized LWE*)

Comme RLWE, mais le secret et l'aléa sont des k-uplets de polynômes dans $Z_q[X]/(X^n + 1)$.

$$\mathcal{E}_{(S_1,\ldots,S_k)}(M) = (A_1, \ldots, A_k, A_1S_1 + \cdots + A_kS_k + \Delta R + M)$$

Unifie LWE (cas $n = 1$) et RLWE (cas $k = 1$).

Encode n nombres avec $kn + n$ nombres.

Addition homomorphe

Chiffrement faiblement homomorphe pour l'addition

La somme point à point de deux chiffrés LWE est le chiffré de la somme modulo t, si le bruit ne déborde pas.

En grand-boutiste :

$$
\begin{aligned}
&(\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta m_1 + r_1) + (\mathbf{a}_2, \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta m_2 + r_2) \\
&\quad = \quad (\mathbf{a}_1 + \mathbf{a}_2, \langle \mathbf{a}_1 + \mathbf{a}_2, \mathbf{s} \rangle + \Delta(m_1 + m_2) + r_1 + r_2) \\
&\quad = \quad \text{un chiffré de } (m_1 + m_2) \bmod t \\
&\qquad\quad\ \ \text{à condition que } |r_1 + r_2| < \Delta/2
\end{aligned}
$$

Chiffrement faiblement homomorphe pour l'addition

Même propriété en petit-boutiste :

$$
\begin{aligned}
&(\mathbf{a}_1, \langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta r_1 + m_1) + (\mathbf{a}_2, \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta r_2 + m_2) \\
&\quad = \quad (\mathbf{a}_1 + \mathbf{a}_2, \langle \mathbf{a}_1 + \mathbf{a}_2, \mathbf{s} \rangle + \Delta(r_1 + r_2) + m_1 + m_2) \\
&\quad = \quad \text{un chiffré de } (m_1 + m_2) \bmod \Delta \\
&\qquad\quad\ \ \text{à condition que } \Delta | r_1 + r_2 + (m_1 + m_2)/\Delta | < q/2
\end{aligned}
$$

Même propriété pour RLWE et GLWE.

Multiplication par une petite constante

Le produit d'un chiffré et d'une constante k est le chiffré du produit du clair par k, si k est assez petite pour éviter le débordement.

En grand-boutiste :

$$\begin{aligned} & k \cdot (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r) \\ = \quad & (k \cdot \mathbf{a}, \langle k \cdot \mathbf{a}, \mathbf{s} \rangle + \Delta(k \cdot m) + k \cdot r) \\ = \quad & \text{un chiffré de } k \cdot m \bmod t \quad \text{si } |k \cdot r| < \Delta/2 \end{aligned}$$

Inutilisable si k est grand par-rapport à Δ.

Multiplication par une constante arbitraire

Soit $q = \beta^\ell$. On peut décomposer la constante k en base β :

$$k = k_0 + k_1\beta + k_2\beta^2 + \cdots + k_{\ell-1}\beta^{\ell-1}$$

On considère un chiffrement redondant Lev (*leveled*) d'un texte clair m mis aux échelles $1, \beta, \beta^2, \ldots$:

$$\mathcal{E}_{Lev}(m) = \big(\mathcal{E}(m), \mathcal{E}(\beta m), \mathcal{E}(\beta^2 m), \ldots, \mathcal{E}(\beta^{\ell-1} m)\big)$$

On peut alors calculer homomorphiquement le produit km :

$$\begin{aligned} & k_0 \cdot \mathcal{E}(m) + k_1 \cdot \mathcal{E}(\beta m) + \cdots + k_{\ell-1} \cdot \mathcal{E}(\beta^{\ell-1} m) \\ = \quad & \text{un chiffré de } k_0 m + k_1 \beta m + \cdots k_{\ell-1}\beta^{\ell-1} m \pmod{q} \\ = \quad & \text{un chiffré de } (k_0 + k_1\beta + \cdots + k_{\ell-1}\beta^{\ell-1})m = km \pmod{q} \end{aligned}$$

Les nombres k_i étant petits, le bruit augmente peu.

Multiplication homomorphe et circuits étagés (approche BGV)

Interlude : rappels sur le produit tensoriel

Le produit tensoriel $\mathbf{u} \otimes \mathbf{v}$ de deux vecteurs de dimensions respectives n et m est le vecteur de dimension nm défini par

$$\mathbf{u} \otimes \mathbf{v} = (u_1 v_1, \ldots, u_1 v_m, \ldots, u_n v_1, \ldots, u_n v_m)$$

Le produit tensoriel commute avec le produit scalaire :

$$\langle \mathbf{u}_1, \mathbf{u}_2 \rangle \cdot \langle \mathbf{v}_1, \mathbf{v}_2 \rangle = \langle \mathbf{u}_1 \otimes \mathbf{v}_1, \mathbf{u}_2 \otimes \mathbf{v}_2 \rangle$$

Multiplication de deux chiffrés (approche BGV)

(Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan, *(Leveled) fully homomorphic encryption without bootstrapping*, ICTS 2012, TOCT 2014)

On considère deux chiffrés LWE de m_1 et m_2 (petit-boutistes) :

$$c_1 = (\mathbf{a}_1, b_1) \qquad \text{avec } \; b_1 = \langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta r_1 + m_1$$
$$c_2 = (\mathbf{a}_2, b_2) \qquad \text{avec } \; b_2 = \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta r_2 + m_2$$

On a

$$\begin{aligned} b_1 \cdot b_2 &= (\langle \mathbf{a}_1, \mathbf{s} \rangle + \Delta r_1 + m_1) \cdot (\langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta r_2 + m_2) \\ &= \langle \mathbf{a}_1, \mathbf{s} \rangle \cdot \langle \mathbf{a}_2, \mathbf{s} \rangle + \Delta(\cdots) + m_1 \cdot m_2 \\ &= \langle \mathbf{a}_1 \otimes \mathbf{a}_2, \mathbf{s} \otimes \mathbf{s} \rangle + \Delta(\cdots) + m_1 \cdot m_2 \end{aligned}$$

Donc, $(\mathbf{a}_1 \otimes \mathbf{a}_2, b_1 \cdot b_2)$ ressemble à un chiffré de $m_1 \cdot m_2$ mais avec la clé $\mathbf{s} \otimes \mathbf{s}$ au lieu de $\mathbf{s}$, et la dimension n^2 au lieu de n.

Multiplication de deux chiffrés (approche BGV)

Chiffrés $(\mathbf{a}_1, b_1)$, $(\mathbf{a}_2, b_2)$

(clé $\mathbf{s}$, dimension n, module q, bruit B)

$\downarrow$ produit tensoriel

Chiffré $(\mathbf{a}_1 \otimes \mathbf{a}_2, b_1 \cdot b_2)$

(clé $\mathbf{s} \otimes \mathbf{s}$, dimension n^2, module q, bruit B^2)

$\downarrow$ changement de clé

Chiffré $(\mathbf{a}', b')$

(clé $\mathbf{s}$ ou $\mathbf{s}'$, dimension n, module q, bruit B^2)

$\downarrow$ réduction de module

Chiffré $(\mathbf{a}'', b'')$

(clé $\mathbf{s}$ ou $\mathbf{s}'$, dimension n, module q/B, bruit B)

Changement de clé

On a un chiffré $c = (\mathbf{a}, b)$ du message m avec la clé $\mathbf{s}$.

On voudrait un chiffré c' de m avec une autre clé $\mathbf{s}'$.

Idée : réaliser un déchiffrement partiel homomorphe de c.

On transmet $\mathbf{s}$ chiffré avec $\mathbf{s}'$ au calculateur :

$$\mathcal{E}_{\mathbf{s}'}(s_1), \ldots, \mathcal{E}_{\mathbf{s}'}(s_n) \quad \text{(sous forme \textit{leveled} Lev)}$$

On peut alors calculer $b - \langle \mathbf{a}, \mathbf{s} \rangle$ de manière homomorphe :

$$\begin{aligned} & \mathcal{E}_0(b) - a_1 \cdot \mathcal{E}_{\mathbf{s}'}(s_1) - \cdots - a_n \cdot \mathcal{E}_{\mathbf{s}'}(s_n) \\ = \quad & \text{un chiffré de } b - \langle \mathbf{a}, \mathbf{s} \rangle \text{ avec la clé } \mathbf{s}' \end{aligned}$$

(Il faut utiliser la forme *leveled* car les facteurs a_i sont grands.)

Changement de clé

En représentation petit-boutiste, $b - \langle \mathbf{a}, \mathbf{s} \rangle = \Delta r + m$.

On a obtenu un chiffrement $(\mathbf{a}', b')$ de $\Delta r + m$ avec la clé $\mathbf{s}'$. Or,

$$\begin{aligned}(\mathbf{a}', b') &= (\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + \Delta r' + (\Delta r + m)) \\ &= (\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + \Delta(r' + r) + m) \\ &= \text{un chiffré de } m \text{ avec } \mathbf{s}' \text{ si } |r' + r| \text{ est assez petit}\end{aligned}$$

Donc,

$$\mathcal{E}_0(b) - a_1 \cdot \mathcal{E}_{\mathbf{s}'}(s_1) - \cdots - a_n \cdot \mathcal{E}_{\mathbf{s}'}(s_n)$$

est un chiffré de m avec la nouvelle clé $\mathbf{s}'$, si le bruit n'augmente pas trop.

Analyse du bruit

Le bruit d'un chiffré est l'ordre de grandeur de Δr.
Il doit rester $< q$ si on veut pouvoir déchiffrer.

Si les deux arguments ont un bruit B, le produit tensoriel puis le changement de clé produisent un chiffré de bruit $\approx B^2$.

Cela limite beaucoup la profondeur multiplicative des circuits et le degré des polynômes évaluables homomorphiquement.
Exemple avec $q = B^{10}$:

	Bruit / Module
chiffré initial	B/B^{10}
profondeur 1, degré 2	B^2/B^{10}
profondeur 2, degré 4	B^4/B^{10}
profondeur 3, degré 8	B^8/B^{10}
profondeur 4, degré 16	erreur ! B^{16}/B^{10}

Changement de module

Idée : réduire le module d'un facteur $\approx B$ à chaque multiplication.

Ainsi, l'erreur absolue reste à peu près constante.
Le facteur limitant devient la taille du module.

Bruit / Module	module constant	réduction de module
chiffré initial	B/B^{10}	B/B^{10}
profondeur 1, degré 2	B^2/B^{10}	$B^2/B^{10} = B/B^9$
profondeur 2, degré 4	B^4/B^{10}	$B^2/B^9 = B/B^8$
profondeur 3, degré 8	B^8/B^{10}	$B^2/B^8 = B/B^7$
profondeur 4, degré 16	erreur! B^{16}/B^{10}	$B^2/B^7 = B/B^6$

Si $q_0 \approx B^n$, on a donc une profondeur maximale de $n - \mathit{constante}$, au lieu de $\log_2 n$ précédemment.

Changement de module (grand-boutiste)

Soit q et q' deux modules. Supposons $\Delta' = \Delta q'/q$ entier.

On considère un chiffré $(\mathbf{a}, b)$ modulo q de m en grand-boutiste :

$$b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r \pmod{q}$$

On définit

$$(\mathbf{a}', b') = \left(\lfloor \mathbf{a}\frac{q'}{q} \rceil, \lfloor b\frac{q'}{q} \rceil\right)$$

En supposant $\mathbf{s}$ binaire (coefficients 0 ou 1), on montre que

$$b' = \langle \mathbf{a}', \mathbf{s} \rangle + \Delta' m + \lfloor r\frac{q'}{q} \rceil + \varepsilon \pmod{q'}$$

avec $|\varepsilon| = \mathcal{O}(n)$.

Si Δ' reste assez grand, $(\mathbf{a}', b')$ est un chiffré modulo q' de m.

Chiffrement faiblement homomorphe étagé

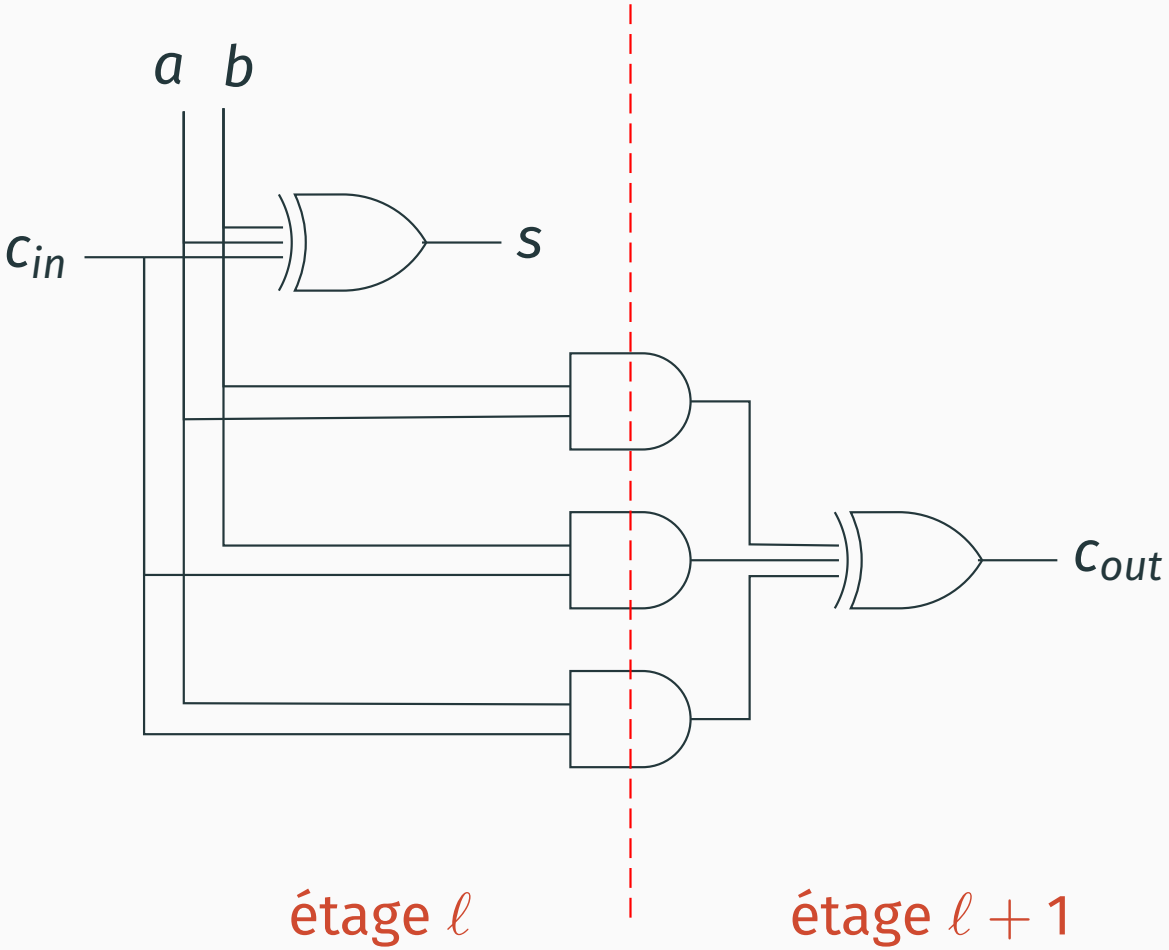
On se donne une profondeur multiplicative maximale d.

Le client choisit des modules $q_1, \ldots, q_d$ avec $q_i/q_{i+1} \approx B$ (B niveau de bruit minimal) et des clés secrètes $\mathbf{s}_1, \ldots, \mathbf{s}_d$.

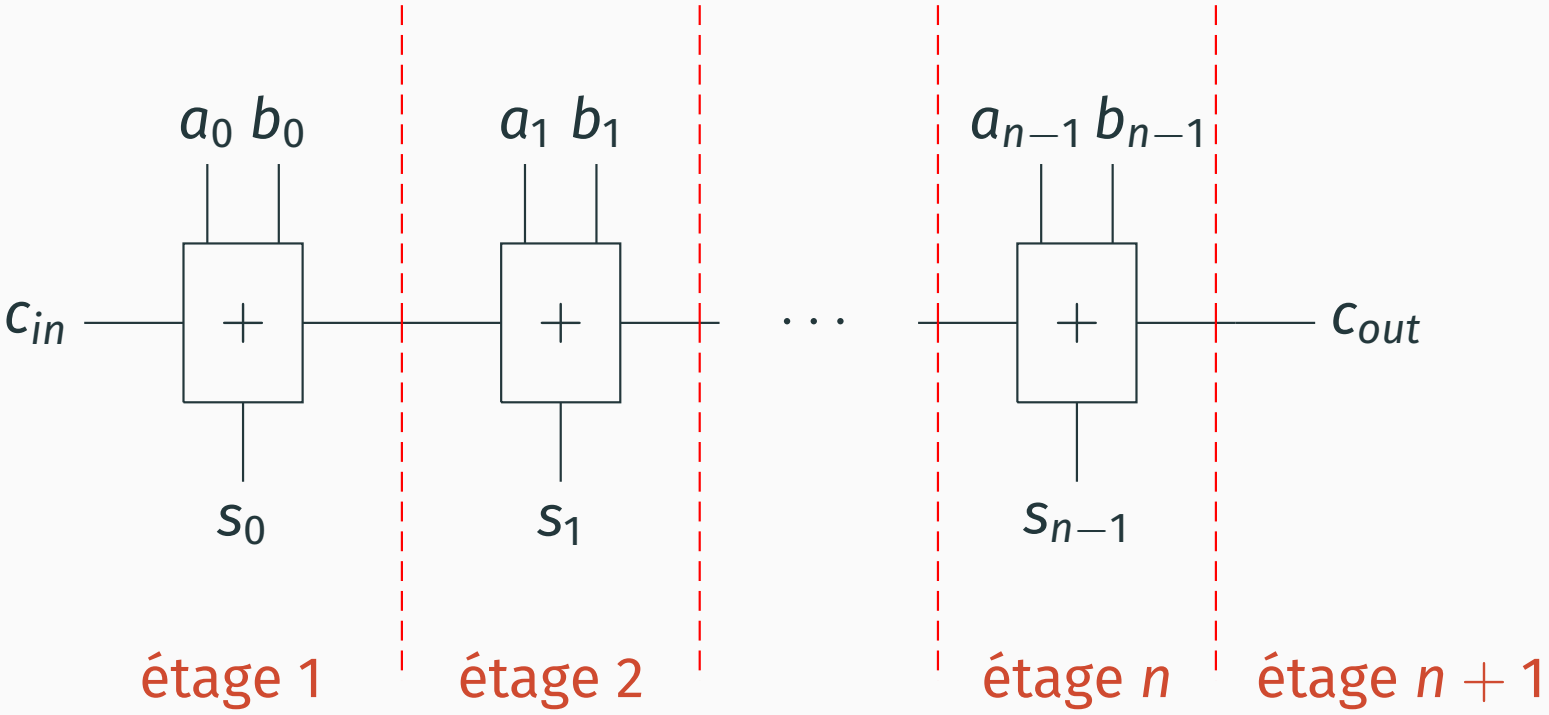
Il transmet au calculateur les q_i et les matrices de changement de clé $\mathcal{E}_{\mathbf{s}_{i+1}}(\mathbf{s}_i \otimes \mathbf{s}_i)$.

Le calculateur peut évaluer homomorphiquement tout circuit de profondeur multiplicative $\leq d$, en changeant de clé et de module à chaque multiplication.

Exemple de circuit étagé : l'additionneur complet



Exemple de circuit étagé : l'additionneur n bits



Bootstrap programmable dans l'approche FHEW/TFHE

Bootstrap vs. changement de clé

Objectif du bootstrap : étant donné un chiffré c avec la clé $\mathbf{s}$, produire un chiffré c' avec la clé $\mathbf{s}'$ (possiblement $= \mathbf{s}$) tel que

1. c' se déchiffre en le même message m que c ;
2. le bruit de c' est réinitialisé et indépendant de celui de c.

Le changement de clé garantit (1) mais pas (2)
(le bruit de c' est le bruit de c + le bruit du changement).

Bootstrap = changement de clé + indexation dans une table

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption : Bootstrapping in less than 0.1 Seconds*, 2016)

Chiffrement grand-boutiste : $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Déchiffrement du chiffré $(\mathbf{a}, b)$:

1. Calculer $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$
2. Extraire $m = \lfloor e/\Delta \rceil$

Bootstrap = changement de clé + indexation dans une table

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption : Bootstrapping in less than 0.1 Seconds*, 2016)

Chiffrement grand-boutiste : $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Déchiffrement homomorphe du chiffré $(\mathbf{a}, b)$:

1. Calculer $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$ chiffré avec $\mathbf{s}'$
2. Extraire $m = \lfloor e/\Delta \rceil$ chiffré avec $\mathbf{s}'$ et un bruit minimal

Bootstrap = changement de clé + indexation dans une table

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption : Bootstrapping in less than 0.1 Seconds*, 2016)

Chiffrement grand-boutiste : $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Déchiffrement homomorphe du chiffré $(\mathbf{a}, b)$:

1. Calculer $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$ chiffré avec $\mathbf{s}'$
2. Extraire $m = \lfloor e/\Delta \rceil$ chiffré avec $\mathbf{s}'$ et un bruit minimal

(1) s'effectue par changement de clé, en utilisant $\mathcal{E}_{\mathbf{s}'}(\mathbf{s})$.

Bootstrap = changement de clé + indexation dans une table

(Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, Malika Izabachène, *Faster Fully Homomorphic Encryption : Bootstrapping in less than 0.1 Seconds*, 2016)

Chiffrement grand-boutiste : $b = \langle \mathbf{a}, \mathbf{s} \rangle + \Delta m + r$.

Déchiffrement homomorphe du chiffré $(\mathbf{a}, b)$:

1. Calculer $e = b - \langle \mathbf{a}, \mathbf{s} \rangle$ chiffré avec $\mathbf{s}'$
2. Extraire $m = \lfloor e/\Delta \rceil$ chiffré avec $\mathbf{s}'$ et un bruit minimal

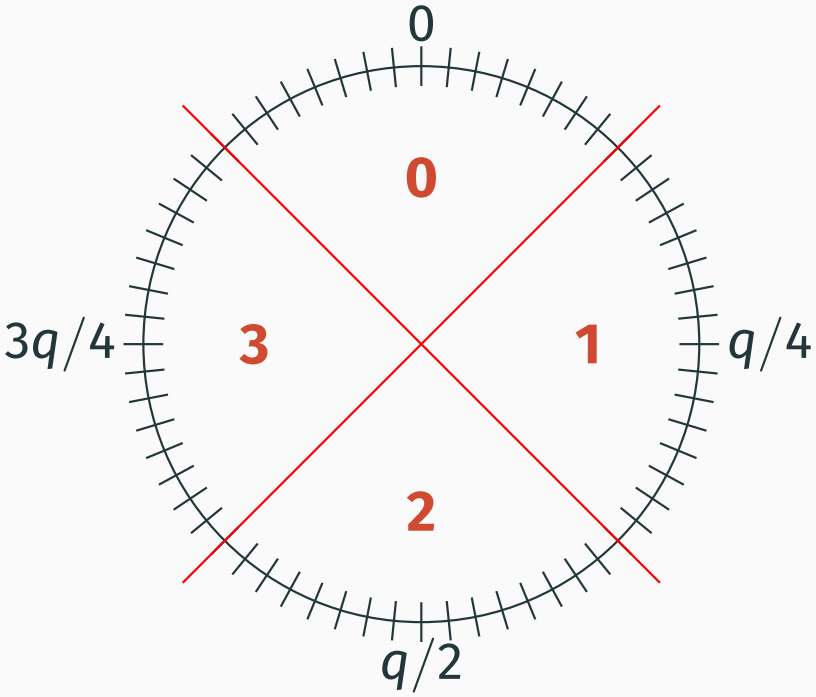
(1) s'effectue par changement de clé, en utilisant $\mathcal{E}_{\mathbf{s}'}(\mathbf{s})$.

(2) s'effectue par indexation homomorphe dans une table

$$i \in \mathbb{Z}_q \longmapsto \text{un chiffré de } \lfloor i/\Delta \rceil \text{ avec } \mathbf{s}' \text{ de bruit minimal}$$

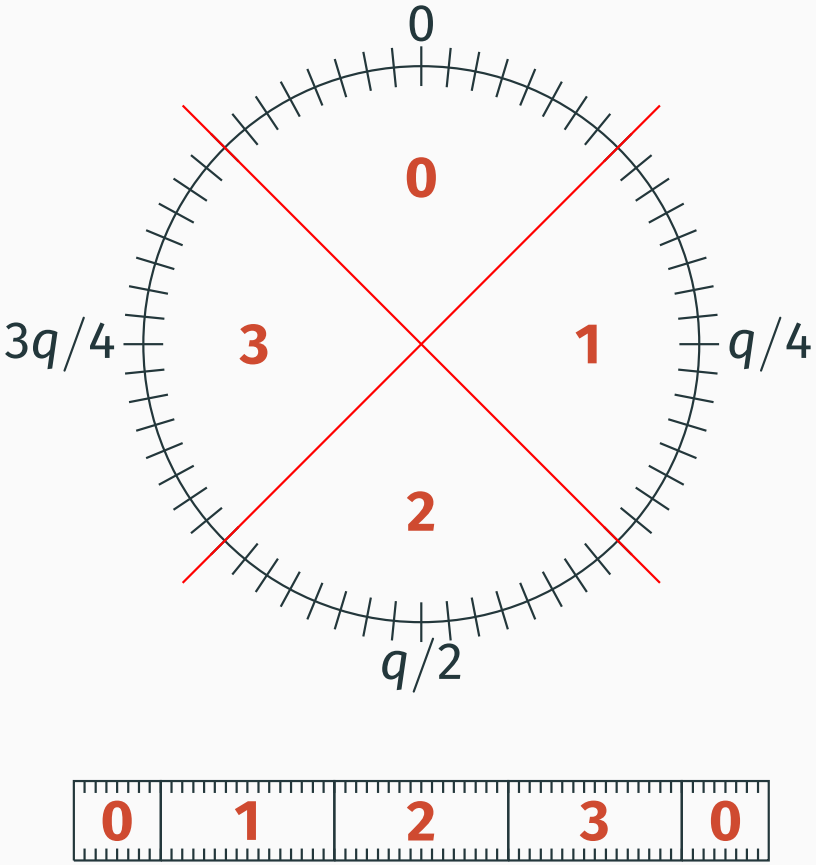
Tores et tables

Cas $q = 64$ et $\Delta = 4$:

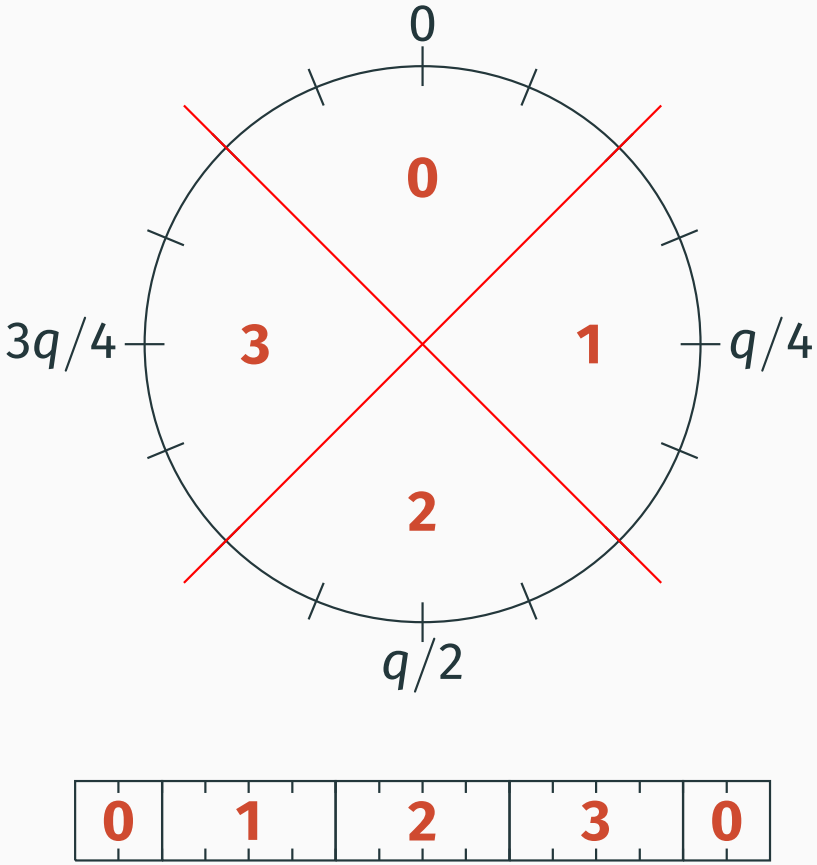


Tores et tables

Cas $q = 64$ et $\Delta = 4$:



Réduction à $q = 16$:



On peut réduire la taille de la table en effectuant au préalable une réduction de module.

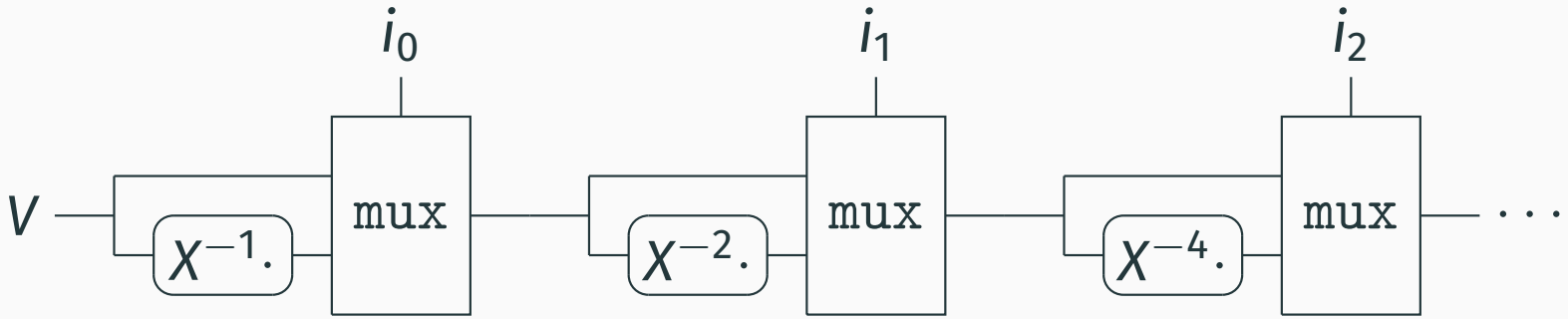
Indexation dans une table

Une table à N entrées est représentée par un polynôme V modulo $X^N + 1$.

Pour extraire l'entrée $i \in [0, N)$,

- on calcule $X^{-i} \cdot V$ (rotation de i cases vers les poids faibles)
- on extrait le coefficient constant de $X^{-i} \cdot V$.

Si on a une décomposition binaire de $i = i_0 + 2i_1 + \cdots + 2^n i_n$, on peut utiliser un décaleur à barillet (*barrel shifter*) :



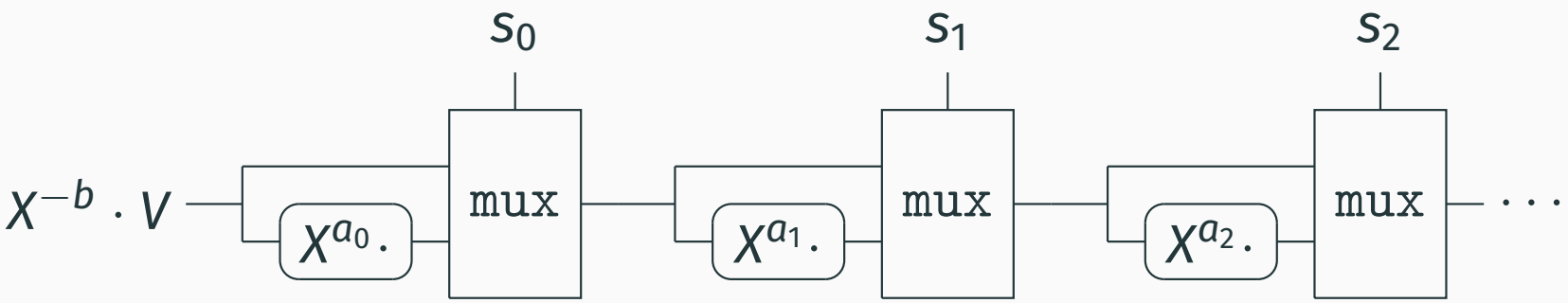
avec $\mathtt{mux}(i, a, b) = a$ si $i = 0$ et $= b$ si $i = 1$.

Application au déchiffrement homomorphe

Après réduction de module si nécessaire, le chiffré est $(\mathbf{a}, b)$ et l'index dans la table V est $i = b - \langle \mathbf{a}, \mathbf{s} \rangle$.

On veut multiplier V par $X^{-i} = X^{-b+a_0 s_0 + \cdots + a_{n-1} s_{n-1}}$.

On suppose que les composantes s_i de la clé sont des bits 0/1. On adapte le décaleur à barillet :



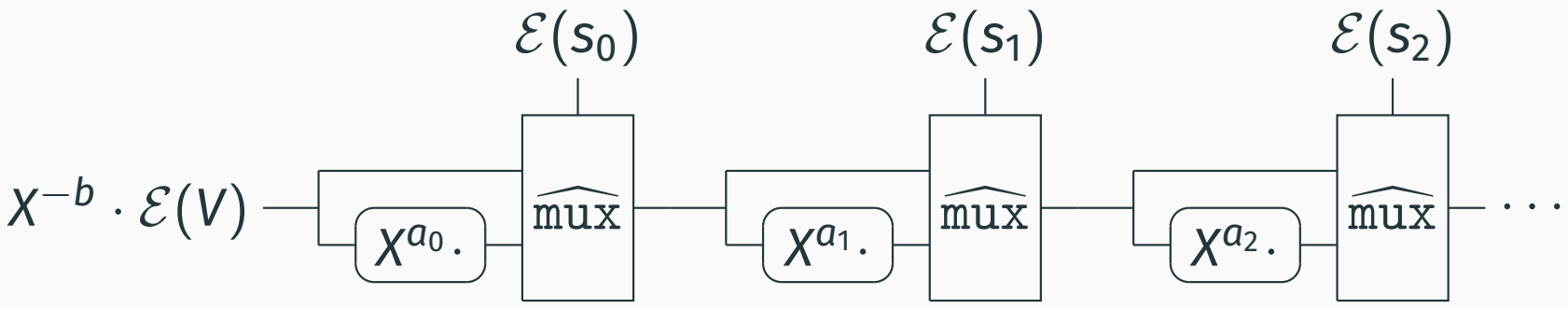
Application au déchiffrement homomorphe

La porte $\texttt{mux}$ peut s'évaluer homomorphiquement :

$$\widehat{\texttt{mux}}(i, a, b) = i \mathbin{\hat{\times}} (b \mathbin{\hat{-}} a) \mathbin{\hat{+}} a$$

où $\hat{+}$, $\hat{-}$, $\hat{\times}$ sont l'addition, la soustraction et la multiplication de deux chiffrés.

On peut donc évaluer le décaleur homomorphiquement :



Le point $\cdot$ est la multiplication clair par chiffré.
Les $\mathcal{E}(s_i)$ sont les chiffrés des bits de la clé $\mathbf{s}$.

Bootstrap accompli !

En sortie du décaleur, on obtient

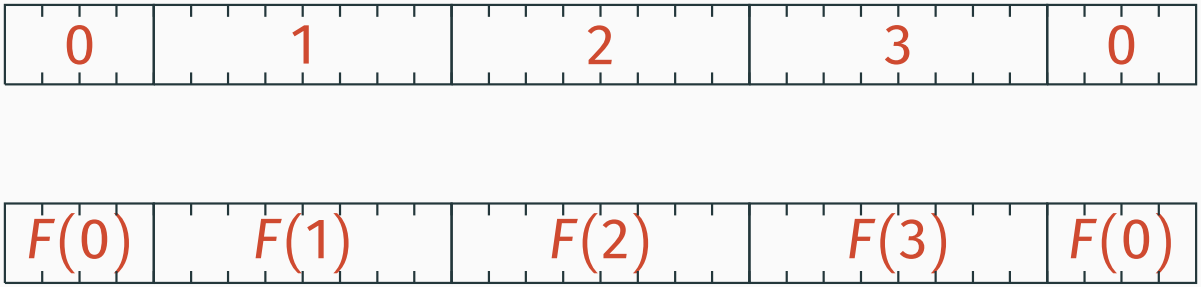
$$X^{-b+a_0\cdot\mathcal{E}(s_0)+\cdots+a_{n-1}\cdot\mathcal{E}(s_{n-1})}\cdot\mathcal{E}(V)$$

dont on peut extraire le coefficient constant, qui est $\mathcal{E}(v_i)$
avec $i = b - \langle \mathbf{a}, \mathbf{s} \rangle = \Delta m + r$ (le terme de bruit),
et v_i la i-ème entrée de la table V :

$$V = \sum_{i=0}^{N-1} v_i X^i \quad \text{avec} \quad v_i = \lfloor i/\Delta \rceil$$

Le résultat est donc un chiffrement peu bruité de $\mathcal{E}(m)$.

Bootstrap programmable



La table V utilisée pour le bootstrap est

$$V = \sum_{i=0}^{N-1} v_i X^i \quad \text{avec} \quad v_i = \lfloor i/\Delta \rceil$$

Mais on peut aussi prendre $v_i = F(\lfloor i/\Delta \rceil)$
pour une fonction $F : \mathbb{Z}_t \to \mathbb{Z}_t$ quelconque.

Alors, la procédure de bootstrap évalue F homomorphiquement tout en réduisant le bruit.

Utilisation du bootstrap programmable

De nombreux calculs se mettent sous la forme d'une fonction unaire appliquée à une combinaison linéaire des entrées.

Minimum et maximum :

$$\begin{aligned} \max(x, y) &= x + \texttt{relu}(y - x) \\ \min(x, y) &= y - \texttt{relu}(y - x) \end{aligned} \qquad \text{avec } \texttt{relu}(z) = \begin{cases} z & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases}$$

Neurone formel :

$$\varphi(w_1 x_1 + \cdots + w_n x_n) \quad \text{où } \varphi \text{ est la fonction d'activation}$$

Multiplication :

$$x \cdot y = F(x + y) - F(x - y) \quad \text{avec } F(z) = z^2/4$$

Point d’étape

Évolutions du chiffrement homomorphe

À partir de la percée conceptuelle mais peu utilisable de Gentry (2009), des progrès majeurs dans trois directions :

- Chiffrement faiblement homomorphe mais de profondeur multiplicative suffisante pour évaluer de nombreux circuits.

  (Brakerski-Gentry-Vaikuntanathan 2011, Brakerski-Fan-Vercauteren 2012, Gentry-Sahai-Waters 2013, etc.)

- Bootstrap efficace et programmable : FHEW, TFHE.

  (Chillotti, Gama, Georgieva, Izabachène, 2016)

  $\Rightarrow$ Séminaire d'Ilaria Chillotti

- Calcul homomorphe approché sur les complexes : CKKS

  (Cheon-Kim-Kim-Song 2016)

  $\Rightarrow$ Séminaire de Damien Stehlé

Mise en pratique

I don't think we'll see anyone using Gentry's solution in our lifetimes.

(Butler Lampson)

Des bibliothèques : HElib, SEAL, TFHE, OpenFHE, HEEAN, ...

(voir https://github.com/jonaschn/awesome-he pour une liste)

Performances non ridicules mais encore limitées sur CPU
(évaluation homomorphe d'un chiffrement AES : 30 secondes).

À l'étude : des implémentations GPU ou matérielles (FPGA, ASIC).

Bibliographie

Bibliographie

Introduction avancée à TFHE :

- Ilaria Chillotti, *TFHE deep dive*, 2022.
  https://www.zama.org/post/tfhe-deep-dive-part-1

Pour référence :

- Ronny Ko, *The Beginner's Textbook for Fully Homomorphic Encryption*, 2025.
  https://arxiv.org/abs/2503.05136 https://fhetextbook.github.io/
- Oded Regev, *The Learning with Errors Problem*, 2010.
  http://www.cims.nyu.edu/~regev/papers/lwesurvey.pdf