Université Paris Cité

Implémentation du cryptosystème original de Gentry

Eytan Levy
Yaël Pajot

Spécialité Cryptologie
Double Master en Mathématiques, Informatique

Semestre 8

Tuteur Université :
Pascal Molin
molin@math.univ-paris-diderot.fr

Université Paris Cité

Résumé

Ce rapport présente une version améliorée du système de chiffrement entièrement homomorphe de Gentry [2], intégrant les améliorations significatives apportées par Gentry et Halevi en 2011 [3]. Ces optimisations simplifient l'exécution complète du processus de cryptage, y compris l'opération cruciale du bootstrapping. De plus, nous détaillons notre propre implémentation, qui utilise le schéma quelque peu homomorphe de Gentry pour effectuer des opérations sur des données chiffrées, tout en analysant ses limites et ses failles de sécurité.

Summary :

This report presents an enhanced version of Gentry's fully homomorphic encryption system [2], incorporating the significant improvements made by Gentry and Halevi in 2011 [3]. These optimizations simplify the complete execution of the encryption process, including the crucial bootstrapping operation. Additionally, we detail our own implementation, which uses Gentry's somewhat homomorphic scheme to perform operations on integers, while analyzing its limitations and security vulnerabilities.

Table des matières

1 Introduction

Le chiffrement homomorphe est un type de chiffrement qui permet d'effectuer des calculs sur des données chiffrées sans avoir besoin d'accéder à la clé privée. On dit qu'un schéma est entièrement homomorphe s'il permet l'évaluation de circuits arbitraires composés de plusieurs types de portes logiques de profondeur illimitée.

1.1 Histoire

Une métaphore pertinente pour illustrer un système de cryptographie homomorphe est celle d'une bijouterie. Prenons l'exemple d'Alice, une bijoutière qui possède des matériaux précieux bruts – or, diamants, argent – qu'elle souhaite voir transformés par ses employés en bagues et colliers élégamment conçus. Toutefois, elle ne fait pas entièrement confiance à ses employés et craint qu'ils ne dérobent les matériaux s'ils en ont l'opportunité. En d'autres termes, elle veut que ses employés puissent travailler sur les matériaux sans pouvoir y accéder directement.

Pour résoudre ce dilemme, Alice conçoit une boîte à gants transparente et inviolable, dont elle seule détient la clé. Elle place les matériaux bruts à l'intérieur de la boîte, la verrouille, et la remet à un employé. Grâce aux gants intégrés, l'employé peut assembler les bijoux à l'intérieur de la boîte, mais comme elle est inviolable, il ne peut pas toucher aux matériaux. Une fois le travail terminé, il rend la boîte à Alice, qui peut alors la déverrouiller avec sa clé et récupérer la bague ou le collier fini.

Dans cette métaphore, les matériaux représentent les données à traiter, tandis que la boîte symbolise le chiffrement de ces données. Ce qui rend cette boîte particulière, en plus d'être inviolable comme d'autres systèmes cryptographiques sécurisés, est qu'elle permet de manipuler les données sans les exposer directement.

En résumé, cette analogie démontre comment le chiffrement homomorphe permet d'exécuter des opérations sur des données chiffrées sans les déchiffrer, garantissant ainsi la sécurité et la confidentialité des informations tout au long du processus.

Le concept de chiffrement homomorphe a été introduit pour la première fois en 1978 par Rivest [1], peu de temps après la découverte de la cryptographie asymétrique et la publication du schéma RSA. ElGamal et RSA sont des cryptosystèmes partiellement homomorphes, c'est-à-dire qu'ils ne permettent d'évaluer qu'un seul type d'opération (addition ou multiplication) à la fois. Cependant, pendant plus de 30 ans, il est resté incertain qu'un schéma de chiffrement entièrement homomorphe puisse être construit.

En 2009, Gentry découvre le premier cryptosystème entièrement homomorphe supportant à la fois l'addition et la multiplication [2], que nous désignerons par « cryptosystème original de Gentry » dans ce rapport. Il commence par construire un

schéma quelque peu homomorphe, capable d'évaluer des polynômes de faible degré sur des données chiffrées, auquel il applique une transformation appelée bootstrapping. Cette transformation permet d'obtenir un schéma pouvant évaluer des polynômes de degré arbitraire tout en réduisant la complexité du déchiffrement des données. Lorsque le degré des polynômes évaluables par le schéma dépasse le double du degré du polynôme de déchiffrement, le schéma est dit «bootstrappable» et peut alors être converti en un schéma entièrement homomorphe.

1.2 Aperçu du projet

Ce rapport se divise en deux grandes sections. La première partie offre une approche théorique du cryptosystème original de Gentry[2], décrivant le schéma de base ainsi que les améliorations apportées par Gentry et Halevi en 2011 [**?**]. La seconde partie se concentre sur notre propre implémentation du schéma quelque peu homomorphe, en examinant ses limites et les applications possibles sur les données.

2 Prérequis

2.1 Notations

Dans ce rapport, les notations suivantes seront utilisées :

— Nous utilisons «$\cdot$» pour désigner la multiplication scalaire et «$\times$» pour toute autre forme de multiplication.
— Pour les entiers z et d, nous notons la réduction de z modulo d par soit $[z]_d$ soit $\langle z \rangle_d$.
  — Nous employons $[z]_d$ lorsque l'opération réduit les entiers à l'intervalle $[-d/2, d/2)$.
  — Nous employons $\langle z \rangle_d$ lorsque l'opération réduit les entiers à l'intervalle $[0, d)$.
  — Nous utilisons la notation générique $z \mod d$ lorsque l'intervalle spécifique n'a pas d'importance (par exemple, mod 2).

Par exemple :

$$[13]_5 = -2 \quad \text{alors que} \quad \langle 13 \rangle_5 = 3$$

$$[9]_7 = \langle 9 \rangle_7 = 2$$

Pour un nombre rationnel q, nous notons par $\lceil q \rfloor$ l'arrondi de q à l'entier le plus proche. Nous notons par $[q]$ la distance entre q et l'entier le plus proche. Ainsi, si $q = \frac{a}{b}$, alors $[q] = \frac{[a]_b}{b}$.

L'arrondi se fait de manière coordonnée pour un vecteur rationnel. Par exemple, si $\vec{q} = \langle q_0, q_1, \ldots, q_{n-1} \rangle$ est un vecteur rationnel, alors l'arrondi se fait pour chaque

coordonnée :

$$\lceil \vec{q} \rfloor = \langle \lceil q_0 \rfloor, \lceil q_1 \rfloor, \ldots, \lceil q_{n-1} \rfloor \rangle$$

.

2.2 Cryptosystèmes basés sur les réseaux

2.2.1 Réseaux

Soit $B = \{\vec{b_0}, \ldots, \vec{b_{n-1}}\}$ une famille de n vecteurs linéairements indépendants sur $\mathbb{R}^m$. Alors le réseau engendré par B est l'ensemble de toutes les combinaisons linéaires entières des vecteurs dans B : $L(B) = \left\{ \sum_i x_i \vec{b_i} \mid x_i \in \mathbb{Z} \right\}$. B est appelée « base du réseau », et celui-ci peut-être représenté par la matrice $\mathbf{B} \in \mathbb{R}^{n \times m}$, dont les colonnes sont les vecteurs $\vec{b_i}$. Alors on peut simplifier la définition de $L(B)$ ainsi :

$$L(B) = \{\vec{x} \times \mathbf{B} : \vec{x} \in \mathbb{Z}^n\}. \tag{1}$$

Dans ce projet on considérera uniquement des réseaux de rang plein, où $n = m$. De plus les coefficients de B sont des entiers ($\vec{b_i} \in \mathbb{Z}^n$).

Deux matrices B_1 et B_2 sont des bases du même réseau L si et seulement s'il existe une matrice unimodulaire U telle que $B_1 = U \times B_2$. Comme U est unimodulaire, $|\det(B_i)|$ est invariant pour différentes bases de L, et nous pouvons le désigner par $\det(L)$. À la base B du réseau L, nous associons le parallélépipède semi-ouvert $\mathcal{P}(B) \leftarrow \{\sum_{i=1}^{n} x_i \vec{b_i} : x_i \in [-1/2, 1/2)\}$. Le volume de $\mathcal{P}(B)$ est précisément $\det(L)$.

Chaque réseau de rang complet possède une forme normale d'Hermite (FNH) unique. Dans cette forme, les éléments $b_{i,j}$ sont égaux à zéro pour tout $i < j$ (formant ainsi une matrice triangulaire inférieure), les $b_{j,j}$ sont strictement positifs pour tout j, et pour tout $i > j$, $b_{i,j}$ appartient à l'intervalle $[-b_{j,j}/2, +b_{j,j}/2)$. La forme normale d'Hermite peut être obtenu en appliquant uniquement des transformations unimodulaires, si bien qu'elle est une base du réseau. La FNH est souvent considérée comme la base qui révèle le moins d'informations sur L, ce qui en fait une représentation typique pour la clé publique du réseau.

SVP et BDD : Le problème γ-SVP (Shortest Vector Problem with Approximation Factor γ) consiste à trouver un vecteur non nul dans un réseau L dont la longueur est au plus γ fois la longueur du plus court vecteur non nul de L. Plus précisément, étant donné un réseau $L \subset \mathbb{R}^n$ avec $\lambda_1(L)$ représentant la longueur du plus court vecteur non nul, le problème γ-SVP est de trouver un vecteur $\mathbf{v} \in L \setminus \{0\}$ tel que $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(L)$. Pour des valeurs élevées de γ ($\gamma = 2^{\Omega(n)}$), l'algorithme de Lenstra-Lenstra-Lovász (LLL) peut trouver une solution en temps polynomial. Un problème ouvert est de savoir s'il existe des algorithmes pour résoudre le SVP exact

en temps exponentiel simple ($2^{O(n)}$) et nécessitant une mémoire évoluant polynomialement en fonction de la dimension du réseau.

Le problème de décodage à distance bornée (BDD) consiste à trouver le point du réseau L le plus proche d'un vecteur donné $\mathbf{c}$ qui est très proche d'un point du réseau. Plus précisément, étant donné un réseau $L \subset \mathbb{R}^n$ et un vecteur $\mathbf{c} \in \mathbb{R}^n$, le but est de trouver un vecteur $\mathbf{v} \in L$ tel que $\|\mathbf{c} - \mathbf{v}\|$ soit minimal. Dans la version γ-BDD, il y a un paramètre $\gamma > 1$ et la promesse que $\|\mathbf{c} - \mathbf{v}\| \leq \frac{\lambda_1(L)}{\gamma}$, où $\lambda_1(L)$ est la longueur du plus court vecteur non nul de L. Comparé à SVP, le problème BDD a reçu beaucoup moins d'attention du point de vue de la théorie de la complexité. De manière générale, les meilleurs algorithmes pour résoudre le γ-BDDP dans des réseaux de dimension n nécessitent un temps exponentiel en $n/\log\gamma$. Plus précisément, les algorithmes actuels peuvent résoudre le γ-BDDP de dimension n en un temps de l'ordre de $2^{\mu n}$ pour $\gamma = 2^{k/\log k}$, où μ est un paramètre dépendant des détails spécifiques de l'algorithme (en extrapolant à partir des expériences de Gama-Nguyen, nous nous attendons à ce que μ soit dans l'intervalle $[0.1, 0.2]$).

2.2.2 Réseaux idéaux

Soit $f(x)$ un polynôme monique irréductible dans $\mathbb{Z}[x]$ de degré n. Soit R l'anneau des polynômes entiers modulo $f(x)$, c'est-à-dire $R = \mathbb{Z}[x]/(f(x))$. Chaque élément de R est un polynôme de degré au plus $n-1$ et peut donc être associé à un vecteur de coefficients dans $\mathbb{Z}^n$. De cette manière, nous pouvons considérer chaque élément de R à la fois comme un polynôme et comme un vecteur. Pour un polynôme $\vec{v}(x)$, la norme euclidienne de son vecteur de coefficients est notée $\|\vec{v}\|$. Pour chaque anneau R, il existe un facteur d'expansion $\gamma_{Mult}(R)$ tel que $\|\vec{u} \times \vec{v}\| \leq \gamma_{Mult}(R) \cdot \|\vec{u}\| \cdot \|\vec{v}\|$, où $\times$ désigne la multiplication dans l'anneau. Lorsque $f(x) = x^n + 1$, comme dans ce rapport, $\gamma_{Mult}(R)$ est $\sqrt{n}$. Cependant, pour des "vecteurs aléatoires" $\vec{u}$ et $\vec{v}$, le facteur d'expansion est généralement beaucoup plus petit et nos expériences suggèrent que $\|\vec{u} \times \vec{v}\| \approx \|\vec{u}\| \cdot \|\vec{v}\|$.

Soit I un idéal de R. Puisque I est fermé sous l'addition, les vecteurs de coefficients associés aux éléments de I forment un réseau. Les idéaux possèdent une structure additive en tant que réseaux, mais également une structure multiplicative. Le produit IJ de deux idéaux I et J est la fermeture additive de l'ensemble $\{\vec{v} \times \vec{w} : \vec{v} \in I, \vec{w} \in J\}$, où «×» représente la multiplication dans l'anneau. Pour simplifier, nous utiliserons des idéaux principaux de R, c'est-à-dire des idéaux ayant un seul générateur. L'idéal $(\vec{v})$ engendré par $\vec{v} \in R$ correspond au réseau engendré par les vecteurs $\{\vec{v}_i = \vec{v} \times x^i \mod f(x) : i \in [0, n-1]\}$; nous appelons cela la base de rotation du réseau idéal $(\vec{v})$.

Soit K la clôture algebrique de R. Sans notre cas $K = \mathbb{Q}[x]/(f(x))$. Alors l'inverse d'un idéal $I \subseteq R$ est $I^{-1} = \{\vec{w} \in K : \forall \vec{v} \in I, \vec{v} \times \vec{w} \in R\}$. Ainsi l'inverse de l'idéal principal $(\vec{v})$ est $(\vec{v}^{-1})$ où $\vec{v}^{-1} \in K$.

2.2.3 Cryptosystèmes GGH

Le premier cryptosystème GGH (Goldreich–Goldwasser–Halevi) est publié en 1997, est un système asymetrique basé sur les réseaux évoqués précédemment. Les clés secrètes et publiques sont respectivement une "bonne" et une "mauvaise" base d'un même réseau L. Plus concrètement, le détenteur de la clé génère une bonne base en choisissant B_{sk} comme une base de vecteurs courts et "presque orthogonaux". Ensuite, il définit la clé publique comme étant la forme normale d'Hermite du même réseau, soit $B_{pk} = \mathrm{HNF}(L(B_{sk}))$, étant la base la moins révélatrice.

Pour chiffrer un message, on l'encode dans un vecteur d'erreur très petit $\vec{e}$ de la même dimension que le réseau, puis on l'ajoute à un point aléatoire du réseau, ce qui donne $\vec{c} = \vec{x}B' + \vec{e}$, où $\vec{x}$ est un vecteur entier aléatoire. Lorsque l'on réduit $\vec{c}$ modulo une "bonne" base B, étant donné que $\vec{e}$ est court, le point de coordonnées $\vec{e}$ se trouve à l'intérieur du parallélépipède $\mathcal{P}(B)$ et peut donc être récupéré comme $\vec{e} = \vec{c} \mod B$. La longueur maximale que peut avoir $\vec{e}$ tout en assurant que $\vec{e} = \vec{c} \mod B$ est appelée le rayon de déchiffrement de B.

Si l'on utilise une "mauvaise" base B', le parallélépipède $\mathcal{P}(B')$ étant beaucoup plus mince que $\mathcal{P}(B)$, son rayon de déchiffrement est beaucoup plus faible que celui de B. Dans ce cas, il n'est plus assuré que $\vec{e} = \vec{c} \mod B$. En fait, trouver le vecteur d'erreur $\vec{e}$ en utilisant une "mauvaise" base se révèle être un problème difficile, ce qui explique pourquoi le chiffrement basé sur les réseaux est sécurisé. Je vais maintenant aborder le cryptosystème homomorphe de Gentry qui est basé sur les réseaux.

2.2.4 Le cryptosystème quelque peu homomorphe

Les deux prochaines sections decrivent le cryptosystème original de Gentry (amélioré grâce au variant Smart-Vercauteren) qui se partage en deux parties : le cryptosystème quelque peu homomorphe, qui permet d'effectuer un nombre limité d'opérations sur des données chiffrées, et le système complètement homomorphe, construit à partir de celui-ci, permet de réaliser un "déchiffrement homomorphe" d'un texte chiffré. Ce déchiffrement homomorphe est ensuite utilisé pour effectuer des calculs arbitrairement longs sur des textes chiffrés. Dans cette section, je décris le schéma partiellement homomorphe.

Le schéma quelque peu homomorphe peut être vu comme un cryptosystème GGH, utilisant les réseaux idéaux. On utilise les réseaux d'idéaux car ces ensembles assurent la closure additive mais aussi multiplicative, ce qui permet d'appliquer additions et multiplications sur nos textes chiffrés. On travaille sur l'anneau $\mathrm{R} = \mathbb{Z}[x]/(f_n(x))$, où $f_n(x) = x^n + 1$ avec n une puissance de 2. Ainsi, notre réseau idéal, que l'on nomme J, est un idéal de l'anneau R. J est un idéal principal définit en choisissant un vecteur aléatoire $\vec{v}$ de degré n, sous certaines conditions auxquelles on reviendra par la suite. Alors $J = (\vec{v})$. La clé publique du cryptosystème, la "mauvaise base", est la forme normale de Hermite de la base de rotation $\mathbf{V}$:

$$\mathbf{V} = \begin{bmatrix} v_0 & v_1 & v_2 & \dots & v_{n-2} & v_{n-1} \\ -v_{n-1} & v_0 & v_1 & \dots & v_{n-3} & v_{n-2} \\ -v_{n-2} & v_1 & v_0 & \dots & v_{n-4} & v_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -v_3 & v_{n-3} & v_{n-4} & \dots & v_0 & v_1 \\ -v_2 & v_{n-2} & v_{n-3} & \dots & v_1 & v_0 \\ -v_1 & v_{n-1} & v_{n-2} & \dots & v_2 & v_3 \end{bmatrix} \tag{2}$$

Plus précisemment, la FNH de cette matrice est :

$$\mathbf{HNF}(\mathbf{J}) = \begin{bmatrix} d & 0 & 0 & \dots & 0 & 0 \\ -r & 1 & 0 & \dots & 0 & 0 \\ -[r^2]_d & 0 & 1 & \dots & 0 & 0 \\ -[r^3]_d & 0 & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -[r^{(n-1)}]_d & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \tag{3}$$

où $d = det(\mathbf{V})$ et r est une racine de $f_n(x)$ modulo d. Ainsi on peut garder seulement d et r comme clé publique, car ces deux entiers seuls permettent de retrouver $HNF(J)$.

Pour chiffrer un vecteur $\langle m, 0, 0, \dots, 0\rangle$ avec $m \in \{0, 1\}$, on peut choisir un polynôme aléatoire à petits coefficients $u(x)$ et évaluer ce polynôme en r. Le chiffré est alors obtenu en calculant l'entier $c \leftarrow [2u(r) + m]_d$.

Smart et Vercauteren ont également décrit une procédure de déchiffrement qui utilise un seul entier w comme clé secrète. Le message est obtenu en calculant $m \leftarrow [\mathrm{cw}]_d \mod 2$.

2.2.5 Le cryptosystème entièrement homomorphe

Le cryptosystème decrit ci-dessus permet d'évaluer des polynômes de faibles degrés sur les données chiffrées, mais lorsque les chiffrés sortent du rayon de dechiffrement, les messages clairs associés ne peuvent plus être retrouvés. Gentry résout ce problème en 2009, et donne une solution pour rendre le schéma bootstrappable. On dit qu'un schéma est bootstrappable s'il est capable d'évaluer homomorphiquement les deux fonctions suivantes :

$$D_{\mathrm{Add}}(\vec{c_1}, \vec{c_2}, sk) = \mathrm{Dec}_{sk}(\vec{c_1}) + \mathrm{Dec}_{sk}(\vec{c_2}) \tag{4}$$

$$D_{\mathrm{Mul}}(\vec{c_1}, \vec{c_2}, sk) = \mathrm{Dec}_{sk}(\vec{c_1}) \times \mathrm{Dec}_{sk}(\vec{c_2}) \tag{5}$$

pour n'importe quels chiffrés $\vec{c_1}$ et $\vec{c_2}$. Un schéma capable de celà peut être transformé en schéma entièrement homomorphe en ajoutant à la clé publique un chiffrement de la clé privée $\vec{c}^* \leftarrow \text{Enc}_{pk}(sk)$. Ainsi, l'addition et la multiplication de deux chiffrés $\vec{c_1}$ et $\vec{c_2}$ peut se faire homomorphiquement en évaluant les fonctions $D_{\text{Add}}(\vec{c_1}, \vec{c_2}, \vec{c}^*)$ ou $D_{\text{Mul}}(\vec{c_1}, \vec{c_2}, \vec{c}^*)$. Ainsi, le bruit n'augmente pas avec le nombre d'opération, et le schéma permet d'évaluer des polynômes de degré arbitraire.

Néanmoins, le schéma quelque peu homomorphe de Gentry n'est pas bootstrappable. En effet, le degré la fonction de dechiffrement exprimé en la taille de la clé privée est trop élevé. Gentry propose alors une mise à jour : celle ci inclut dans la clé publique un "indice" sur la clé privée $\vec{w}$ - un ensemble $\mathbf{S}$ de S vecteurs $\vec{x_i}$ qui contient un sous-ensemble caché T dont la somme est égale à $\vec{w}$. L'ancienne clé privée, $\vec{w}$, est remplacée par le vecteur caractéristique du sous-ensemble T, qui est noté $\vec{\sigma} = \langle \sigma_1, \sigma_2, \ldots, \sigma_S \rangle$. Dans le nouveau schéma, la complexité du dechiffrement s'exprime comme un polynôme en les σ_i de degré approximativement égal à la taille du sous-ensemble parcimonieux T. En définissant astucieusement les parametres, la taille de T peut être assez petite et permettre au schéma d'être bootstrappable.

Le schéma quelque peu homomorphe

3 Génération des clés

La génération des clés consiste à générer aléatoirement un réseau idéal principal $(\vec{v})$ de l'anneau de polynômes $\mathbb{Z}[x]/(f_n(x))$, où $f_n(x) = x^n + 1$ avec n une puissance de 2, en vérifiant que sa forme normale d'Hermite est de la forme (3) ci-dessus. Les parametres à prendre en compte lors de cette phase est n, la dimension de notre réseau, et la taille t des coefficients du polynôme générateur $\vec{v}$ de l'idéal principal (en nombre de bits).

— Choisissons au hasard un réseau entier n-dimensionnel $\vec{v}$, où chaque entrée v_i est choisie aléatoirement comme un entier signé de t bits. À ce vecteur $\vec{v}$, nous associons le polynôme formel $v(x) = \sum_{i=0}^{n-1} v_i x^i$, ainsi que la base de rotation $\mathbf{V}$ illustrée dans (2) ci-dessus.

— On calcule $w(x)$ ainsi :

$$w(x) \times v(x) = \text{constante} \pmod{f_n(x)} \tag{6}$$

On remarquera par la suite que la constante est égale à $\det(L(B))$. On désigne cette constante par d, et on dénote les coefficients de $w(x)$ par $\vec{w} = \langle w_0, w_1, \ldots, w_{n-1} \rangle$. Il sera aisé de remarquer que la base de rotation $\mathbf{W}$ de $\vec{w}$ est l'inverse de $\mathbf{V}$, et que $\mathbf{W} \times \mathbf{V} = \mathbf{V} \times \mathbf{W} = d \cdot \mathbf{I}$.

— Enfin on vérifie que le polynôme générateur convient au cryptosystème, c'est à dire que sa forme normale d'Hermite est de la bonne forme. Il a été observé

par Nigel Smart que la forme normale d'Hermite (HNF) est correcte lorsque le déterminant est impair et sans facteur carré. Cependant le deuxième critère est difficile à vérifier, et on decrit ci-dessous une procédure permettant de vérifier la matrice en calculant l'inverse.

3.1 Vérification de la FNH

En 2011, Gentry et Halevi prouvent que la forme normale d'Hermite du réseau $L(V)$ a la bonne forme si et seulement si le réseau contient un vecteur de la forme $\langle -r, 1, 0, \ldots, 0\rangle$. Ainsi, si la matrice est de la bonne forme, il existe un vecteur $\vec{y}$ et un entier r tel que :

$$\vec{y} \times V = \langle -r, 1, 0, \ldots, 0\rangle \tag{7}$$

En multipliant à gauche et à droite par W, on obtient :

$$\vec{y} \times (dI) = -r \cdot \langle w_0, w_1, w_2, \ldots, w_{n-1}\rangle + \langle -w_{n-1}, w_0, w_1, \ldots, w_{n-2}\rangle$$

$$\Leftrightarrow \dot{d\vec{y}} = -r \cdot \langle w_0, w_1, w_2, \ldots, w_{n-1}\rangle + \langle -w_{n-1}, w_0, w_1, \ldots, w_{n-2}\rangle$$

En réduisant cela modulo d, cela signifie :

$$-r \cdot \langle w_0, w_1, \ldots, w_{n-1}\rangle + \langle -w_{n-1}, w_0, \ldots, w_{n-2}\rangle = 0 \pmod{d}$$

$$\Leftrightarrow \langle -w_{n-1}, w_0, \ldots, w_{n-2}\rangle = r \cdot \langle w_0, w_1, \ldots, w_{n-1}\rangle \pmod{d}.$$

Cette dernière condition peut être facilement vérifiée : nous calculons $r := \frac{w_0}{w_1} \mod d$ (en supposant que w_1 possède un inverse modulo d), puis nous vérifions que $r \cdot w_{i+1} = w_i \pmod{d}$ pour tout $i = 1, \ldots, n-2$ et également que $-r \cdot w_0 = w_{n-1} \pmod{d}$. Notez que cela signifie en particulier que $r^n = -1 \pmod{d}$. (Dans notre implémentation, nous vérifions en fait seulement cette dernière condition, au lieu de tester toutes les égalités $r \cdot w_{i+1} = w_i \pmod{d}$).

3.2 Clé privée et clé publique

Il sera plus pertinent, par la suite, de ne garder dans la clé publique que les entiers d et r, qui représentent à eux deux la FNH. De même, en principe la clé privée est le couple (,), mais on verra par la suite qu'en vertu de nos fonctions de chiffrement et de dechiffrement, il suffit de garder comme clé privée un seul coefficient impair de $\vec{w}$.

4 Chiffrement

Soit B la forme normale d'Hermite du réseau engendré par . Pour chiffrer un bit $b \in \{0, 1\}$, nous commençons par choisir aléatoirement un vecteur de bruit

$0, \pm 1$ $\vec{u} = \langle u_0, u_1, \ldots, u_{n-1} \rangle$, chaque entrée étant choisie comme 0 avec une certaine probabilité q et comme ± 1 avec une probabilité $(1-q)/2$ chacune. Ensuite, nous définissons

$$\vec{a} = 2\vec{u} + b \cdot \vec{e}_1 = \langle 2u_0 + b, 2u_1, \ldots, 2u_{n-1} \rangle$$

et le texte chiffré est le vecteur

$$\vec{c} = \vec{a} \mod B = \vec{a} - \left\lfloor \vec{a} \times B^{-1} \right\rceil \times B = \underbrace{\left[\vec{a} \times B^{-1} \right]}_{\text{partie fractionnaire}} \times B \qquad [\cdot]$$

Nous montrons maintenant que $\vec{c}$ peut également être représenté implicitement par un seul entier. Rappelons que B (et donc B^{-1}) ont une forme particulière :

$$B = \begin{pmatrix} d & 0 & 0 & 0 & 0 \\ -r & 1 & 0 & 0 & 0 \\ -[r^2]_d & 0 & 1 & 0 & 0 \\ -[r^3]_d & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -[r^{n-1}]_d & 0 & 0 & 0 & 1 \end{pmatrix}, \quad B^{-1} = \begin{pmatrix} d & 0 & 0 & 0 & 0 \\ r & d & 0 & 0 & 0 \\ [r^2]_d & 0 & d & 0 & 0 \\ [r^3]_d & 0 & 0 & d & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ [r^{n-1}]_d & 0 & 0 & 0 & d \end{pmatrix}.$$

Nous notons $\vec{a} = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ et désignons par $a(\cdot)$ le polynôme entier $a(x) = \sum_{i=0}^{n-1} a_i x^i$. Ainsi, nous avons $\vec{a} \times B^{-1} = \langle s, a_1, \ldots, a_{n-1} \rangle$ pour un certain entier s qui satisfait $s = a(r) \pmod d$.

La partie fractionnaire de $\vec{a} \times B^{-1}$ est donc $[\vec{a} \times B^{-1}] = [a(r)]_d \langle 1, 0, \ldots, 0 \rangle$ et le vecteur de texte chiffré est

$$\vec{c} = [a(r)]_d \langle 1, 0, \ldots, 0 \rangle \times B = \langle [a(r)]_d, 0, \ldots, 0 \rangle.$$

Il est clair que ce vecteur peut être représenté par un seul entier $c = [a(r)]_d$.

Pour chiffrer b, il suffit donc évaluer le polynôme de bruit $u(\cdot)$ en r, de le multiplier par deux et d'ajouter le bit b (le tout modulo d).

4.1 Parametrer le bruit

Pour choisir un vecteur de bruit pour un nouveau texte chiffré, il est nécessaire de maximiser la probabilité q que chaque élément soit nul, tout en maintenant q suffisamment inférieur à 1 pour éviter la récupération facile du vecteur de bruit original. Les attaques par réduction de réseau peuvent être contrées en utilisant des réseaux de haute dimension, nous nous concentrons donc sur les attaques par recherche exhaustive. Pour un vecteur de bruit avec l bits d'entropie, les attaques de type "birthday" (l'attaquant tente de trouver deux messages d'entrée différents qui produisent la même valeur de hachage) peuvent le récupérer en temps $2^{l/2}$. Par conséquent, pour un paramètre de sécurité λ, nous devons assurer que le bruit ait au moins 2λ bits d'entropie, ce qui implique que q soit suffisamment petit pour que

$2^{(1-q)n} \cdot n > 2^{2\lambda}$. Une attaque hybride pourrait consister à choisir un sous-ensemble aléatoire de puissances de r pour inclure tous les coefficients de bruit. Par exemple, en travaillant avec une dimension $n = 2048$ et seulement 16 entrées non nulles pour le bruit, nous aurions une probabilité d'environ $(200/2048)^{16} \approx 2^{-54}$ d'inclure toutes les entrées. Pour nos défis publics, nous avons opté pour un paramètre agressif avec le nombre d'entrées non nulles fixé à 15, notant que l'augmentation du bruit n'augmente que modérément la taille de la clé et le temps de calcul.

5 Dechiffrement

La procédure de déchiffrement prend le texte chiffré c (qui représente implicitement le vecteur $\vec{c} = \langle c, 0, \ldots, 0 \rangle$) et utilise en principe les deux matrices V et W. Le vecteur $\vec{a} = 2\vec{u} + b \cdot \vec{e_1}$, utilisé pendant le chiffrement, est récupéré comme suit :

$$\vec{a} \leftarrow \vec{c} \mod V = \vec{c} - \left(\left[\vec{c} \times V^{-1}\right] \times V\right) = \left[\vec{c} \times \frac{W}{d}\right] \times V,$$

et ensuite la procédure retourne le bit de poids faible de la première entrée de $\vec{a}$, c'est-à-dire $b := a_0 \mod 2$.

Cette méthode de déchiffrement fonctionne car les lignes de V (et donc aussi de W) sont presque orthogonales entre elles, ce qui rend la norme l_∞ de W petite. En effet, pour tout vecteur $\vec{x}$, la plus grande entrée de $\vec{x} \times W$ (en valeur absolue) n'est pas beaucoup plus grande que la plus grande entrée de $\vec{x}$ lui-même. Plus précisément, la procédure réussit lorsque toutes les entrées de $\vec{a} \times W$ sont inférieures à $d/2$ en valeur absolue. Pour comprendre cela, remarquons que $\vec{a}$ est la distance entre $\vec{c}$ et un point dans le réseau $L(V)$, c'est-à-dire que nous pouvons exprimer $\vec{c}$ comme $\vec{c} = \vec{y} \times V + \vec{a}$ pour un vecteur entier $\vec{y}$.

Ainsi, nous avons :

$$\left[\vec{c} \times \frac{W}{d}\right] \times V = \left[\vec{y} \times V \times \frac{W}{d} + \vec{a} \times \frac{W}{d}\right] = \left[\vec{a} \times \frac{W}{d}\right] \times V$$

où l'égalité est vérifiée car $\vec{y} \times V \times \frac{W}{d}$ est un vecteur entier. Le vecteur $\vec{a} \times \frac{W}{d} \times V$ est censé être $\vec{a}$ lui-même, c'est-à-dire que nous avons besoin de

$$\left[\vec{a} \times \frac{W}{d}\right] \times V = \vec{a} = \left(\vec{a} \times \frac{W}{d}\right) \times V.$$

Cette dernière condition est vérifiée si et seulement si chaque entrée de $\vec{a} \times \frac{W}{d}$ doit être inférieure à $1/2$ en valeur absolue.

5.1 Dechiffrer de manière efficace

Nous allons maintenant montrer que le bit chiffré b peut être récupéré par une procédure beaucoup moins coûteuse. Rappelons que le vecteur de texte chiffré $\vec{c}$ est

déchiffré en bit b lorsque la distance entre $\vec{c}$ et le vecteur le plus proche dans le réseau $L(V)$ est de la forme $\vec{a} = 2\vec{u} + b\vec{e_1}$, et que toutes les entrées de $\vec{a} \times W$ sont inférieures à $d/2$ en valeur absolue. Comme mentionné précédemment, dans ce cas nous avons $[\vec{c} \times W/d] = [\vec{a} \times W/d] = \vec{a} \times W/d$, ce qui équivaut à la condition

$$[\vec{c} \times W]_d = [\vec{a} \times W]_d = \vec{a} \times W.$$

Rappelons maintenant que $\vec{c} = \langle c, 0, \ldots, 0 \rangle$, donc

$$[\vec{c} \times W]_d = [c \cdot \langle w_0, w_1, \ldots, w_{n-1} \rangle]_d = \langle [cw_0]_d, [cw_1]_d, \ldots, [cw_{n-1}]_d \rangle.$$

D'autre part, nous avons

$$[\vec{c} \times W]_d = \vec{a} \times W = 2\vec{u} \times W + b\vec{e_1} \times W = 2\vec{u} \times W + b \cdot \langle w_0, w_1, \ldots, w_{n-1} \rangle.$$

En combinant ces deux équations, nous obtenons que tout texte chiffré déchiffrable c doit satisfaire la relation

$$\langle [cw_0]_d, [cw_1]_d, \ldots, [cw_{n-1}]_d \rangle = b \cdot \langle w_0, w_1, \ldots, w_{n-1} \rangle \pmod 2.$$

En d'autres termes, pour chaque i, nous avons $[c \cdot w_i]_d = b \cdot w_i \pmod 2$. Il suffit donc de garder seulement l'un des w_i (qui doit être impair), puis de récupérer le bit b comme $b := [c \cdot w_i]_d \pmod 2$.

6 Elargir aux entiers

Avec le cryptosystème decrit ci-dessus, nous pouvons aisément chiffrer, opérer et dechiffrer des bits. Il est notamment possible d'élargir l'espace des messages clairs, par exemple à l'ensemble $\{0, 1, \ldots, p-1\}$. Ainsi, au lieu de choisir le coefficient w impair, on impose que $w = 1 \pmod p$, et les fonctions de chiffrement et dechiffrement deviennent :

$$\mathrm{Enc}(m, p) = \left[m + p \sum_{i=0}^{r} u_i \right]_d$$
$$\mathrm{Dec}(c, p) = [cw]_d \mod p$$

De plus, il est obligatoire de vérifier, lors de la génération des clés, que $d \not\equiv 0 \pmod p$.

Le schéma entièrement homomorphe

Rappelons que la procédure de déchiffrement de notre schéma "quelque peu homomorphe" déchiffre un texte chiffré $c \in \mathbb{Z}_d$ en utilisant la clé secrète $w \in \mathbb{Z}_d$ en

définissant $b \leftarrow [wc]_d \mod 2$. Le cryptosystème permet bien d'effectuer des opérations sur des bits chiffrés de la manière suivante :

$$c_1 \oplus c_2 = [c_1 + c_2]_d$$

$$c_1 \wedge c_2 = [c_1 \times c_2]_d \,.$$

Malheureusement, en considérant c et d comme des constantes et en prenant la fonction de déchiffrement $D_{c,d}(w) = [wc]_d \mod 2$, le degré de $D_{c,d}$ (en tant que polynôme en les bits de la clé secrète) est plus élevé que ce que notre schéma quelque peu homomorphe peut gérer. Par conséquent, ce schéma n'est pas encore bootstrappable. Pour atteindre notre objectif, nous modifions donc le format de la clé secrète et ajoutons des informations à la clé publique pour obtenir une routine de déchiffrement de degré inférieur.

7 Amélioration de la procédure de dechiffrement

On ajoute à notre cryptosystème une donnée qui servira d'indice dans la clé publique : il s'agit d'un probleme de somme de sous-ensemble, dont le résultat est le coefficient w (le secret). Étant donné deux paramètres s et S (dans l'implémentation, nous utilisons systématiquement $s = 15$ et S est un grand nombre), nous ajoutons à la clé publique s entiers $x_1, \ldots, x_s$, tous dans $\mathbb{Z}_d$. Chacun de ces entiers x_k représentera un grand ensemble B_k constitué des S entiers $\{\langle x_k \cdot R^i \rangle_d \mid i \in \{0, \ldots, S-1\}\}$, où R est un paramètre du schéma (dans notre implémentation, on fixe $R = 2^{51}$). Nous ferons référence aux éléments de B_k comme $\{x_k(i) \mid i \in \{0, \ldots, S-1\}\}$. Comme mentionné, nous voulons que ces grands ensembles soient une indication sur w. Pour ce faire, nous choisissons nos entiers $x_1, \ldots, x_s$ de manière à ce qu'il y ait un unique indice i_k pour chaque grand ensemble tel que $\sum_{k=1}^{s} x_k(i_k) = w \pmod d$. Cela signifie en particulier que $[cw]_d = [\sum_{k=1}^{s} c x_k(i_k)]_d$.

Nous voulons maintenant une nouvelle clé secrète associée à cette indication que nous pourrons chiffrer et utiliser pour effectuer la recryption. Nous définissons les nouveaux vecteurs de bits $\vec{\sigma}_1, \ldots, \vec{\sigma}_s$, chacun de taille S, tels que :

$$\sigma_k(i) = \begin{cases} 1 & \text{si } i = i_k \\ 0 & \text{sinon} \end{cases}$$

Chacun de ces bits peut ensuite être chiffré individuellement et inclus dans la clé publique. Pour simplifier, nous ferons référence à $\sigma_k(i)$ comme $\sigma_{k,i}$.

On peut utiliser l'indice pour effectuer une partie des calculs dans le clair. Notamment, pour un certain chiffré c, on peut précalculer les $y_{i,j} = \langle c x_i(j) \rangle_d$. La fonction de decryption s'ecrit alors :

$$\begin{aligned}\mathrm{Dec}(c) &= \left[\sum_{i=1}^{s}\sum_{j=0}^{S-1}\sigma_{i,j}y_{i,j}\right]_d \mod 2 \\ &= \left(\sum_{i=1}^{s}\sum_{j=0}^{S-1}\sigma_{i,j}y_{i,j}\right) - d\cdot\left\lfloor\frac{\sum_{i=1}^{s}\sum_{j=0}^{S-1}\sigma_{i,j}y_{i,j}}{d}\right\rfloor \mod 2 \\ &= \left(\sum_{i=1}^{s}\sum_{j=0}^{S-1}\sigma_{i,j}y_{i,j}\right) - d\cdot\left\lfloor\sum_{i=1}^{s}\sum_{j=0}^{S-1}\sigma_{i,j}\frac{y_{i,j}}{d}\right\rfloor \mod 2.\end{aligned}$$

Remarquons que les quotients $y_{i,j}/d$ peuvent également être calculés en clair et que leur valeur est un nombre rationnel dans $[0,1)$. Étant donné un paramètre de précision ρ, considérons l'approximation $z_{i,j}$ de chaque quotient $y_{i,j}/d$ avec ρ bits après la virgule binaire. Il est montré par Gentry et Halevi que si nous maintenons la distance entre c et le réseau en dessous de $1/(s+1)$, alors utiliser $\rho = \lceil \log_2(s+1)\rceil$ fera en sorte que $\lfloor\sum_{i=1}^{s}\sigma_{i,j}z_{i,j}\rfloor$ donnera le même résultat que $\lfloor\sum_{i=1}^{s}\sigma_{i,j}\frac{y_{i,j}}{d}\rfloor$.

La multiplication d'un bit chiffré $\sigma_{i,j}$ par $z_{i,j}$ va produire un vecteur de bits chiffrés de taille $\rho+1$. La valeur représentée par n'importe quel élément de ce vecteur sera simplement 0 lorsque le bit correspondant de $z_{i,j}$ est 0, et $\sigma_{i,j}$ lorsqu'il est 1.En utilisant le modulo 2 dans la formule ci-dessus, nous pouvons transformer le déchiffrement en le calcul de bits suivant :

$$\mathrm{Dec}(c) = \left(\bigoplus_{i=1}^{s}\bigoplus_{j=0}^{S-1}\sigma_{i,j}\langle y_{i,j}\rangle_2\right) \oplus \langle d\rangle_2 \cdot \left\langle\left\lfloor\sum_{i=1}^{s}\bigoplus_{j=0}^{S-1}\sigma_{i,j}z_{i,j}\right\rfloor\right\rangle_2$$

Le XOR des $\sigma_{i,j}z_{i,j}$ correspond simplement au XOR bit à bit du vecteur de bits chiffrés qu'ils génèrent. Il nous reste une seule étape, qui est l'addition des s vecteurs créés par $\bigoplus_{j=0}^{S-1}\sigma_{i,j}z_{i,j}$, et arrondir à l'entier le plus proche. Pour ce faire, il faut implémenter une version homomorphe de la Grade-School Addition, détaillée par Gentry et Halevi.

7.1 Réduire la taille des clés

Gentry et Halevi donnent une méthode pour réduire drastiquement la taille de la clé privée dans notre schéma : sa taille est à la base de $s \times S$ bits, et l'algorithme suivant permet de la compresser en s vecteurs de q bits, nécessitant $\geq \lceil 2S\rceil$ espace en mémoire (ce qui est beaucoup moins), permettre de réduire le degré de la fonction de dechiffrement homomorphe. Voici en quoi la méthode consiste :

— Pour chacun des ensembles B_k, on définit

$$\vec{\eta}_k = \{\eta_{k,i} \mid i \in \{1,\ldots,q\}\},$$

où q est un entier supérieur à $\lceil\sqrt{2S}\rceil$ et tous les bits de $\vec{\eta}_k$ sont nuls sauf deux, que l'on chiffre à l'aide de la fonction d'encryptage. Une clé secrète $\sigma_{k,i}$ peut être retrouvée en multipliant deux de ces chiffrés.

— Soit $\binom{c}{2}$ l'ensemble des paires de nombres distincts dans $[1, c]$. Pour tout $a \neq b \in [1, c]$, désignons par $i(a, b)$ l'indice de la paire (a, b). C'est-à-dire,

$$i(a,b) \stackrel{\text{def}}{=} (a-1)\cdot c - \binom{a}{2} + (b-a).$$

— Alors on configure $\vec{\eta}_k$ de telle sorte que si $\eta_{k,a}$ et $\eta_{k,b}$ sont les deux encryptions de 1 dans $\vec{\eta}_k$, alors $i(a, b) = i_k$.

Soit $\{\eta_m^k : k \in [s], m \in [c]\}$ les bits dont l'encryption est stockée dans la clé publique (où pour chaque k, exactement deux des bits η_m^k sont '1' et les autres sont '0', et chacun des bits $\sigma_{k,i}$ est obtenu comme un produit de deux des η_m^k).

Nous pouvons maintenant résumer la fonction finale utilisée pour le recryptage (noter que $\langle d\rangle_2$ a été supprimé car d est impair) :

$$\begin{aligned}\mathrm{Dec}(c) &= \left(\bigoplus_{i=1}^{s}\bigoplus_{a,b}\eta_{i,a}\eta_{i,b}\left\langle y_{i,\mathrm{ind}(a,b)}\right\rangle_2\right) \oplus \left\langle\left\lceil\sum_{i=1}^{s}\bigoplus_{a,b}\eta_{i,a}\eta_{i,b}z_{i,\mathrm{ind}(a,b)}\right\rfloor\right\rangle_2 \\ &= \left(\bigoplus_{i=1}^{s}\bigoplus_{a}\eta_{i,a}\bigoplus_{b}\eta_{i,b}\left\langle y_{i,\mathrm{ind}(a,b)}\right\rangle_2\right) \oplus \left\langle\left\lceil\sum_{i=1}^{s}\bigoplus_{a}\eta_{i,a}\bigoplus_{b}\eta_{i,b}z_{i,\mathrm{ind}(a,b)}\right\rfloor\right\rangle_2. \end{aligned} \tag{8}$$

Nous devons choisir $c \geq \lceil 2S\rceil$ pour pouvoir encoder n'importe quel indice $i \in [S]$ par une paire $(a, b) \in \binom{c}{2}$, mais nous pouvons choisir c encore plus grand. Augmenter c augmentera la taille de la clé publique en conséquence, mais diminuera le nombre de multiplications nécessaires pour évaluer l'Équation (10). En particulier, fixer $c = \lceil 2\sqrt{S}\rceil$ augmente les exigences d'espace (par rapport à $c = \lceil 2S\rceil$) seulement par un facteur de 2, mais réduit de moitié le nombre de multiplications. En conséquence, dans notre implémentation, nous utilisons le paramètre $c = \lceil 2\sqrt{S}\rceil$.

Implementation

8 Cryptosystème quelque peu homomorphe

8.1 Trouver le bon réseau idéal

On répète les opérations suivantes, jusqu'à trouver un polynôme générateur *vecv* convenable :

— Le polynôme générateur v est généré aléatoirement, et ses coefficients sont $-2^{t-1} \leq v_i < 2^{t-1}$.

— Pour calculer $w(x)$ l'inverse du polynôme w, on se sert de l'algorithme d'Euclide étendu. Cet algorithme s'éxécute en $O(\log^2(w))$, ce qui est convenable ici, mais il est possible d'utiliser une transformation rapide de Fourier pour accélérer ce processus. Cette étape permet notamment de calculer d, et on vérifie qu'il soit congru à 1 modulo notre espace des chiffrés
— La plupart des tests echouent lors de l'étape suivante : il faut vérifier que

$$\Leftrightarrow \langle -w_{n-1}, w_0, \ldots, w_{n-2} \rangle = r \cdot \langle w_0, w_1, \ldots, w_{n-1} \rangle \pmod{d}.$$

Comme expliqué dans la partie 3, il suffit de calculer r, puis de vérifier que r est une racine de $\mathrm{f}_n(x)$.
Le temps de génération d'une clé valide augmente exponentiellement en fonction des parametres n et t.

9 Cryptosystème entièrement homomorphe

9.1 Compression des clef

Pour rendre notre schéma bootstrappable, il est nécessaire de réduire la complexité de notre rechiffrement homomorphe. Après la génération de clé, nous avons implémenté la génération du problème de sous ensemble détaillé dans la **Section 7.1**, afin de remplacer la clé privée de notre schéma quelque peu homomorphe par s vecteurs de taille $\lceil \sqrt{2 \times S} \rceil + 1$. Ici les temps de génération sont très aléatoires, mais augmentent en fonction du nombre de bits du coefficients w.

9.2 Déchiffrement

Il faut pour pouvoir réduire le degré de la fonction de déchiffrement écrire une fonction d'addition utilisable sur S entiers de $\rho + 1$ bits chiffrés, et donc ne se servant que de fonctions booléennes (XOR et AND). Pour ce faire on arrange les bits dans s colonnes de $\rho + 1$ bits. Le bit de cette colonne dans le résultat sera le XOR de tous les éléments de la conne. Les retenues qu'une colonne j enverra à une colonne j+Δ est le polynôme symétrique élémentaire 2^{Δ} évalué en les bits de la colonne j. Il faudra donc déterminer les polynômes symétriques élémentaires de degré $2^1, 2^2$ jusqu'à 2^{Δ}), avec nb le nombre de bits dans la colonne j. Cela peut se faire en $m2^{\Delta}$ d'après Gentry et Halevi [3].

Evaluation

Cette partie du rapport expose les performances du cryptosystème quelque peu homomorphe implémenté.

10 Performances

Nous

10.1 Benchmark

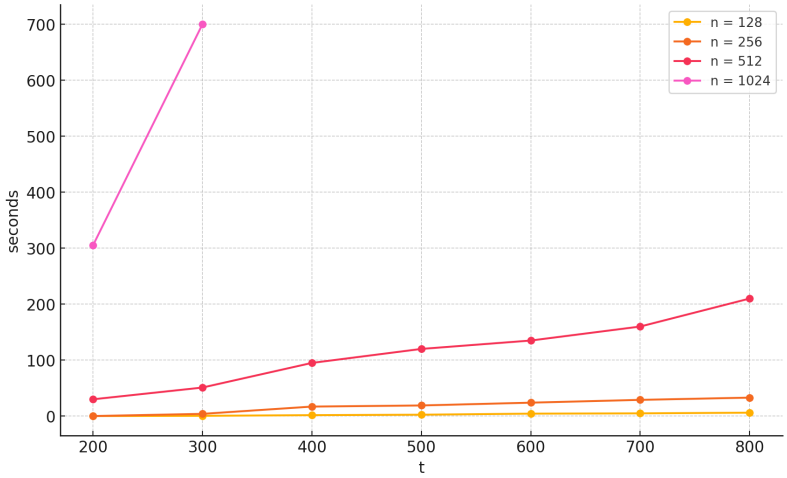

FIGURE 1 – Comparaison du temps de génération de clef en fonction de n et t

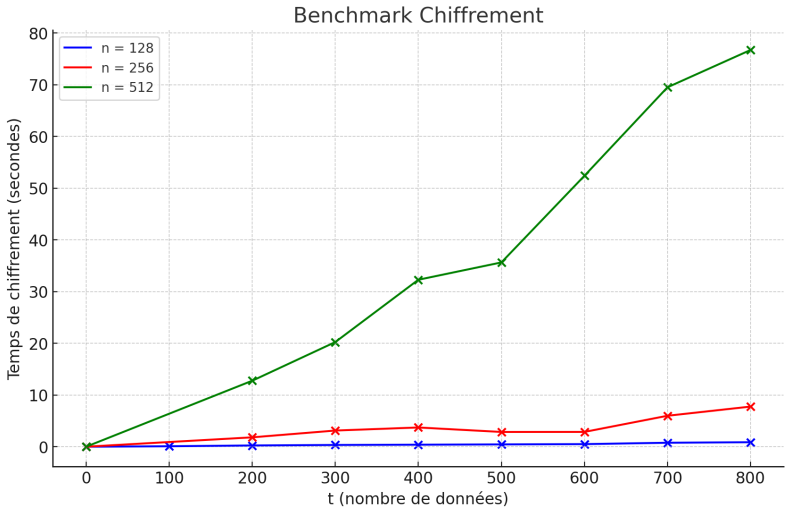

FIGURE 2 – Comparaison du temps de chiffrement en fonction de n et t

Le temps de génération de clef et de chiffrement augmente exponentiellement en fonction de n, et avec le parametre t. En effet, trouver aléatoirement une base valide prend plus de temps lorsque le polynôme augmente de degré. Cependant il

est essentiel de garder ces parametres assez élevés, pour garantir la sécurité de notre schéma.

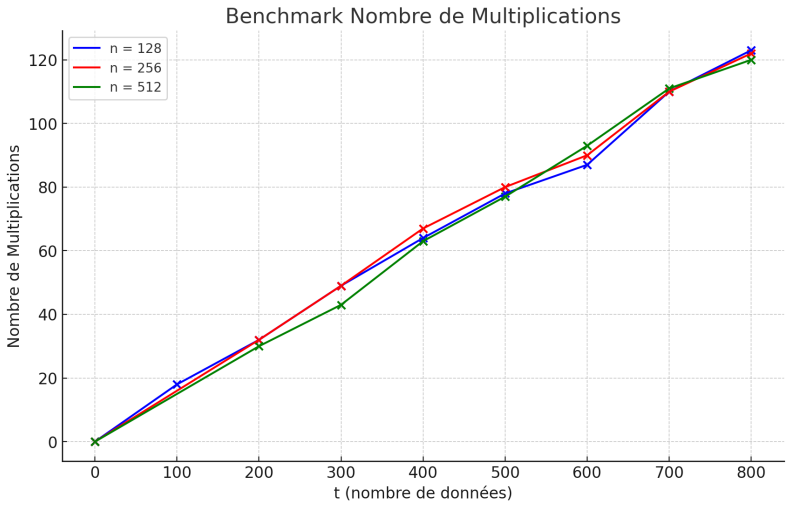

FIGURE 3 – Comparaison du nombre de multiplications avant corruption en fonction de n et t

Le nombre de multiplications possibles sans bootstrapping sur un chiffré augmente linéairement avec la taille en bit des coefficients du polynôme v, mais n'est nullement influencé par la dimension du réseaé ideal.

De plus, il a été observé que le nombre de multiplications possibles sans bootstrapping diminue drastiquement en faisant tendre le paramètre p vers 2. Nous concluons que des fonctions booléennes sur des messages clairs modulo $p >> 2$ fait augmenter exponentiellement le bruit des chiffrés.

Le schéma quelque peu homomorphe s'avère donc être très peu efficace en pratique, ne permettant d'effectuer qu'un nombre limité d'opérations sur de petits nombres. Cependant un schéma bootstrappable garantit l'évaluation de n'importe quelle fonction booléennes, mais est lui aussi très lent.

Il faudra attendre quelques années pour que de nouveaux cryptosystèmes, notamment BGV ou BFV, réduisent encore plus la complexité du rechiffrement homomorphe, le bruit, et permettent au temps d'execution des opération linéaires de diminuer, si bien que pour certains cas d'utilisation, le schéma reste viable sans bootstrapping.

Références

[1] S.L. Graham and R.L. Rivest, *: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, MIT Laboratory for Computer Science and Department of Mathematics, https ://people.csail.mit.edu/rivest/pubs/RSA78.pdf, 1978

[2] Craig Gentry, *: A fully homomorphic encryption scheme*, stanford university, 2009

[3] Craig Gentry and Shai Halevi, *: Implementing Gentry's Fully-Homomorphic Encryption Scheme*, IBM Research, 2010.

[4] Valentin Dalibard, *: Implementing Homomorphic Encryption*, St John's College, 2011.

[5] coron, *: fhe*, https ://github.com/coron/fhe, 2012.