Implémentation d'un Cryptosystème Homorphe par les Réseaux Euclidiens

Reda SOUADI
Nicolas QUERO

Juin 2020

Introduction

Fonctionnement des Cryptosystèmes actuels

Chiffrer des données permet de cacher un message de manière à ce que personne d'autre que ceux disposant de la clé ne puissent le déchiffrer.

Si on souhaite qu'un tiers effectue des opérations sur nos données, il n'est pas possible actuellement de le leur permettre sans qu'ils aient accès à nos données.

Pour répondre à ce besoin, on introduit le chiffrement homomorphe qui a fait de grosses avancées cette dernière décennie notamment grâce aux travaux de *Craig Gentry*.

Chiffrement Homomorphe

Un cryptosystème est dit homomorphe s'il permet d'additionner et de multiplier des chiffrés entre eux de manière à ce qu'après déchiffrement on obtienne le résultat des opérations sur les messages associés.

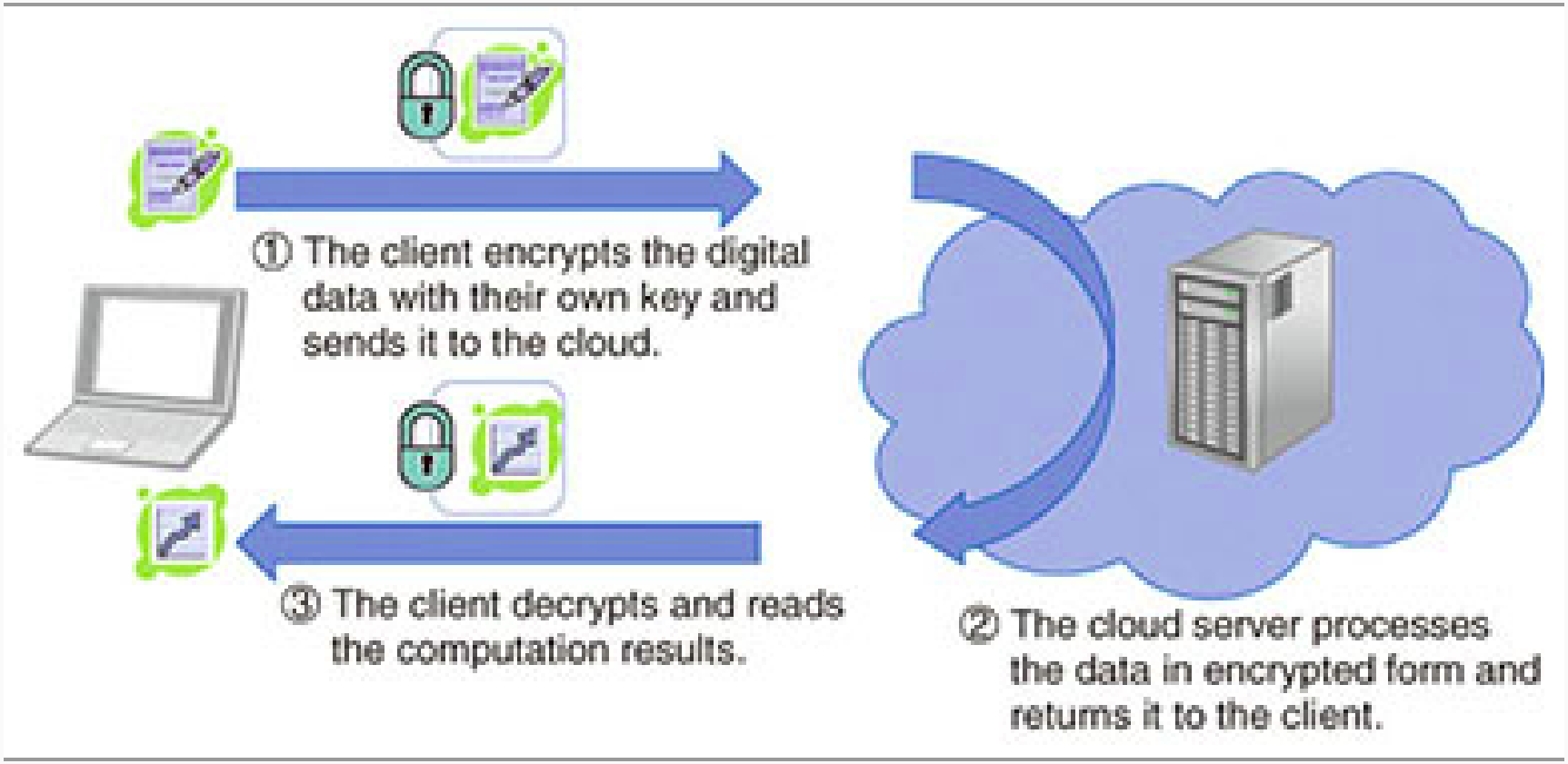
En effet, toute opération logique peut être exprimée comme une suite d'additions et de multiplications.

Quelques applications

L'idée générale est que le chiffrement homomorphe permet d'effectuer des opérations logiques à partir d'entrées chiffrées.

- Cloud computing
- Moteur de recherche aveugle
- Système d'élection chiffré
- Logiciels éxécutables de manière totalement chiffrée

Introduction iv



Chiffrement Homomorphe

Définition

Soit un cryptosystème avec une fonction de chiffrement $enc(m) = c$ et une fonction de déchiffrement $dec(c) = m$.

Le système est dit homomorphe si pour deux messages quelconques $m_1, m_2 \in \mathbf{M}$ et leurs chiffrés respectifs $c_1, c_2 \in \mathbf{C}$ on a :

- $dec(c_1 + c_2) = m_1 + m_2$
- $dec(c_1 \times c_2) = m_1 \times m_2$

Exemple rapide

Considérons l'exemple d'un cryptosystème très simple où $enc(x) = 2x$ et $dec(c) = c/2$.

Regardons ce que donnent la somme et la multiplication dans ce système.

Soit $x_1 = 2$, $x_2 = 5$.
Alors $c_1 = 2x_1 = 4$ et $c_2 = 2x_2 = 10$.
$c_1 + c_2 = 14$ donc $dec(c_1 + c_2) = 14/2 = 7$.
$x_1 + x_2 = 2 + 5 = 7 = dec(c_1 + c_2)$ donc ce système est homomorphe pour l'addition.

Exemple rapide (2)

Maintenant, $c_1 \times c_2 = 4 \times 10 = 40$, $dec(c_1 \times c_2) = 40/2 = 20$.
Mais $x_1 \times x_2 = 2 \times 5 = 10 \neq 20$.

On déduit alors que ce système est homomorphe pour l'addition mais pas pour la multiplication.

D'autres systèmes sont homomorphes pour une seule opération, RSA pour la multiplication par exemple.

SWHE (Somewhat Homomorphic Encryption)

Il est difficile d'avoir un système complètement homomorphe (***Fully Homomorphic***), certains systèmes sont homomorphes jusqu'à effectuer un certain nombre d'additions $(+)$ et de multiplications $(\times)$.

C'est ce qu'on appelle le ***Somewhat Homomorphic Encryption***. Ces systèmes font que le chiffré contient du "bruit" qui grandit à mesure que l'on effectue des opérations. Et lorsque celui-ci devient trop grand, on ne peut plus décrypter correctement le chiffré.

SWHE (Somewhat Homomorphic Encryption)

Ce type de système est au coeur des systèmes les plus prometteurs actuellement, notamment depuis que *Craig Gentry* a introduit la notion de *Bootstrapping* que l'on verra plus loin.

Un premier exemple SWHE

Voyons un cryptosystème dû à Dijk, Gentry, Halevi et Vaikuntanathan. C'est un système **symétrique** donc la clé privée est partagée.

La clé est un grand entier impair p, pour chiffrer un bit $b \in \{0, 1\}$, on tire au hasard deux entiers q et r tels que $r << p$. On chiffre ensuite notre message m par $c = pq + 2r + b$.

Pour déchiffrer, il suffit de faire c mod p mod 2 pour faire disparaître pq puis $2r$.

Un premier exemple SWHE (2)

Voyons ce que donnent les opérations sur des chiffrés c_1, c_2 :

- $$\begin{aligned} c_1 + c_2 &= pq_1 + 2r_1 + b_1 + pq_2 + 2r_2 + b_2 \\ &= p(q_1 + q_2) + 2(r_1 + r_2) + b_1 + b_2 \\ &= pq' + 2r' + b' \end{aligned}$$

  Donc $dec(c_1 + c_2) = c_1 + c_2 \bmod p \bmod 2 = b_1 + b_2$.

- $$\begin{aligned} c_1 \times c_2 &= p(q_1 c_2 + c_1 q_2) + 2(b_1 r_2 + b_2 r_1 + 2r_1 r_2) + b_1 \times b_2 \\ &= pq'' + 2r'' + b_1 \times b_2 \end{aligned}$$

  Donc $dec(c_1 \times c_2) = c_1 \times c_2 \bmod p \bmod 2 = b_1 \times b_2$

Un premier exemple SWHE (3)

Ce système marche donc bien... jusqu'à ce que r se rapproche de la valeur de $(p+1)/2$ par exemple, auquel cas on aura $c = pq + 2(p+1)/2 + b = p(q+1) + 1 + b$.

Et donc $dec(c) = 1 + b \bmod 2$, ce qui est un problème.

Le Bootstrapping

Soit un système **SWHE**, par définition nous pouvons déjà effectuer un nombre fixé d'opérations $(+)$ et $(\times)$.

Dans un tel système, l'idée du Bootstrapping est de recrypter un chiffré (à l'aide d'indices chiffrés sur la clé secrète) avant que le bruit ne soit trop élevé pour permettre de rafraîchir ce bruit et pouvoir réaliser des opérations à nouveau. Le Bootstrapping a un coût et il faut donc que ce coût en opérations soit inférieur au nombre d'opérations possibles.

Chiffrement Homomorphe x

Le système décrit dessus ne sera cependant pas utilisé ici principalement car il demande une clé d'une taille trop conséquente donc nous allons préférer une autre approche

Prérequis

Les réseaux i

Soit $B \in GL_n(\mathbb{R})$, on appelle un réseau L de base B l'ensemble $L = \mathcal{L}(B) = \{Bx | x \in \mathbb{Z}^n\}$

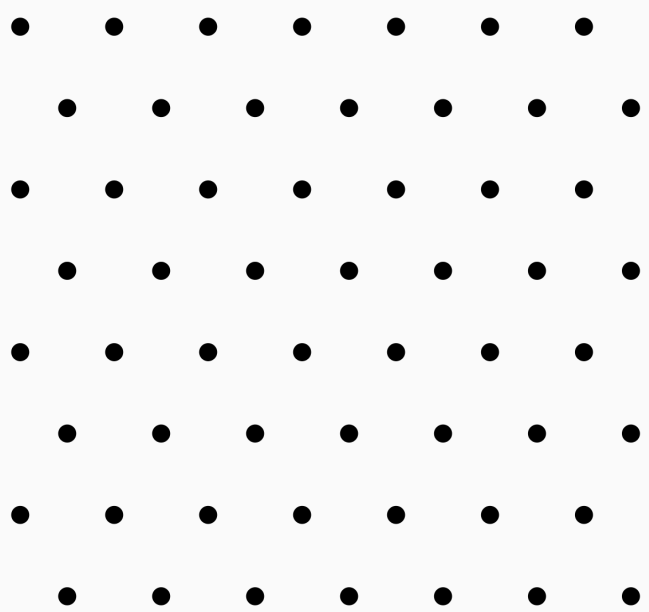


FIGURE 1 – Les points d'un réseau de $\mathbb{R}^2$

Les réseaux ii

Un réseau L a une infinité de bases.

Soit B_1, B_2 deux bases de L il existe une matrice U unimodulaire ($U \in GL_n(\mathbb{Z})$ tq $|det(U)| = 1$) tel que $B_1 = U \times B_2$

Alors $\forall B_1, B_2$ bases de L on a $|det(B_1)| = |det(B_2)|$
On défini donc $det(L) := |det(B)|$ où B est une base de L

Les réseaux iii

On appelle domaine fondamental de L l'ensemble $P(B)$:

$$P(B) = \left\{ \sum_{i=1}^{n} \lambda_i \times b_i, \lambda_i \in \left[\frac{-1}{2}, \frac{1}{2} \right[\right\}$$

Son volume est égal à $det(L)$

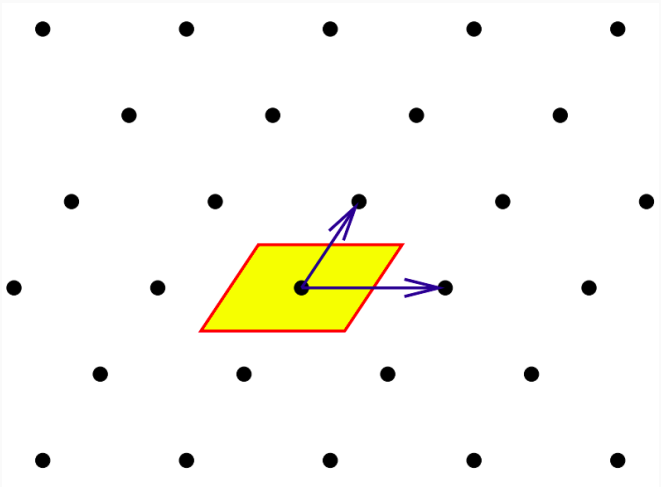
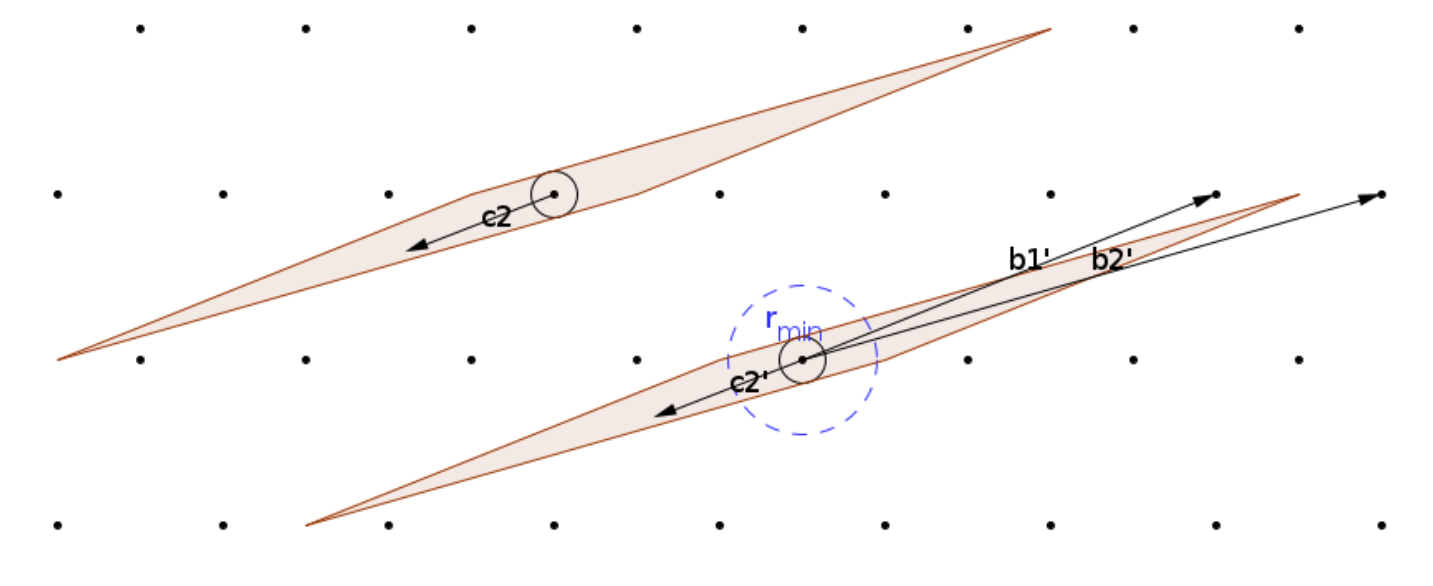


FIGURE 2 – Le domaine fondamental d'un réseau de $\mathbb{R}^2$

Les réseaux iv



Avec une mauvaise base (quasi-colinéaire), le parallélogramme est bien plus étiré donc beaucoup moins de points sont concentrés dans le rayon de décryptage.

Opérations modulo L

Soit B une matrice de L et $v \in \mathbb{R}^n$ il existe un unique vecteur $e \in P(B)$ tel que $v - e \in L$. On note $e : v \bmod B$

Pour calculer e :

$$e = v \bmod B = v - B\lceil B^{-1}v \rfloor$$

Intuitivement ça revient à tracer au tour de chaque point de L le domaine fondamental de B on aura alors un partitionnement de $\mathbb{R}^2$. Il suffit de voir dans quel parallélépipède se trouve v et le point u de L qui correspond à ce parallélépipède, on a donc $e = v - u$

Les réseaux vi

Soit $f \in \mathbb{Z}[X]$ un polynôme irréductible et unitaire de degré n, le quotient $R = \mathbb{Z}[X]/(f)$ est un anneau intègre puisque l'idéal (f) est premier. R contient les polynômes de degré au plus $n - 1$.

Soit I un idéal de R, si on fait correspondre à chaque $g \in I$ un vecteur constitué des coefficients de g alors l'ensemble de ces vecteurs est un réseau de dimension n.

Une base de du réseau de $I = (g)$ est la matrice B constitué des vecteurs $(X^i g(X))_{i \in \{0, n-1\}}$ i.e $B = (g(X), Xg(X), \ldots, X^{n-1}g(X))$

I est donc muni d'une structure de réseau et une structure d'idéal donc l'addition et la multiplication pour lesquels il est stable

Implémentation d'un Cryptosystème par les Réseaux Idéaux

– **Créer la bonne base**

Soit n une puissance de 2, $f_n(x) = X^n + 1$ ($R = \mathbb{Z}[X]/(f_n)$) et $v(x)$ est un polynôme aléatoire de degré n qui est le générateur du réseau.

On remarque que

$$\begin{aligned} v(x) \times x &= (v_0 + v_1 x + \ldots + v_{n-1} x^{n-1}) \cdot x \bmod f_n(x) \\ &= v_0 x + \ldots + v_{n-2} x^{n-1} + v_{n-1} x^n \bmod f_n(x) \\ &= -v_{n-1} + v_0 x + \ldots + v_{n-2} x^{n-1} \bmod f_n(x) \\ &\text{car } x^n + 1 = 0 \text{ donc } x^n = -1 \end{aligned} \tag{1}$$

Implémentation d'un cryptosystème par les réseaux ii

La base est :

$$V = \begin{pmatrix} v_0 & v_1 & \dots & v_{n-1} \\ -v_{n-1} & v_0 & \dots & v_{n-2} \\ \vdots & & \ddots & \vdots \\ -v_1 & -v_2 & \dots & v_0 \end{pmatrix} \tag{2}$$

$$V^{-1} = \frac{1}{d} W \text{ où } W = \begin{pmatrix} w_0 & w_1 & \dots & w_{n-1} \\ -w_{n-1} & w_0 & \dots & w_{n-2} \\ \vdots & & \ddots & \vdots \\ -w_1 & -w_2 & \dots & w_0 \end{pmatrix} \tag{3}$$

chaque ligne de W est le vecteur des coefficients de du polynôme $w(x)$ tel que $w(x) \times v(x) = d = det(V) \bmod f_n(x))$

Implémentation d'un cryptosystème par les réseaux iii

Cette base sera notre **clé secrète**, elle est **quasi-orthogonale**. Il reste à générer la clé publique.

Implémentation d'un cryptosystème par les réseaux iv

– Créer la mauvaise base

Pour créer la mauvaise base, on utilisera la forme normale de Hermite qui est représentable par deux entiers r et d uniquement :

$$B = \begin{pmatrix} d & 0 & 0 & \dots & 0 \\ -r & 1 & 0 & \dots & 0 \\ -[r^2]_d & 0 & 1 & \dots & 0 \\ \vdots & & & \ddots & \\ -[r^{n-1}]_d & 0 & 0 & \dots & 1 \end{pmatrix}, B^{-1} = \frac{1}{d} \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ r & d & 0 & 0 & \dots & 0 \\ [r^2]_d & 0 & d & 0 & \dots & 0 \\ \vdots & & & \ddots & & \\ [r^{n-1}]_d & 0 & 0 & \dots & 0 & d \end{pmatrix} \tag{4}$$

d est le déterminant de V et r est une racine de $f_n(x)$ modulo d. Seules 2 coordonnées diffèrent pour 2 vecteurs distincts, c'est pourquoi cette base est **quasi-colinéaire**.

Chiffrer un bit

Pour chiffrer un bit $b \in \{0, 1\}$ avec B, on choisit un vecteur u à n coordonnées où on en choisira 15 aléatoirement qui seront mises à 1 ou -1, et les autres seront fixées à 0.

Ensuite, on pose $a = (2u_0 + b, 2u_1, ..., 2u_{n-1})$

Puis $c = a \bmod B = [a \times B^{-1}] \times B$

Implémentation d'un cryptosystème par les réseaux vi

$$[a \times B^{-1}] = [(a_0 + a_1 r + a_2 [r^2]_d + ... + a_{n-1}[r^{n-1}]_d,\ a_1 d,\ a_2 d, ...,\ a_{n-1} d) \times \frac{1}{d}]$$
$$= [(\frac{s}{d}, a_1, ..., a_{n-1})] \text{ où } s = \sum_{i=0}^{n-1} a_i [r^i]_d \tag{5}$$

$$c = ([\frac{s}{d}], [a_1], ..., [a_{n-1}]) \times B$$
$$= (\frac{[a(r)]_d}{d}, 0, ..., 0) \times B \text{ car } a_i \in \{-2, 0, 2\}\ \forall i > 0 \text{ donc } a_i \text{ est entier}$$
$$= ([a(r)]_d, 0, ..., 0) \text{ par multiplication par B} \tag{6}$$

Donc un chiffré est représenté par $a(r) \bmod d = [b + 2\sum_{i=0}^{n-1} u_i r^i]_d$

Déchiffrer un bit

Pour déchiffrer un chiffré c il suffit de calculer c mod B_{sk} le vecteur calculé est a. En effet au lieu de faire un produit matriciel il suffit de calculer

$$dec(c) = [c \times w_i]_d$$

où w_i est un coefficient de du polynôme $w(x)$ tel que
$w(x) \times v(x) = d = det(V) \bmod f_n(x)$

Si c n'est plus dans le parallélépipède donc ne peut plus être déchiffre correctement le résultat du calcul est un vecteur qui est très diffèrent de a avec des coefficients très grands.

Fonctionnement des opérations dans ce système i

Dans le système que l'on a défini, nous pouvons déjà effectuer un certain nombre d'opérations car nous sommes dans un idéal donc $(+)$ et $(\times)$ sont des opérations internes.

Additionner des chiffrés

En additionnant deux chiffrés cela donne

$$\begin{aligned} c_1 + c_2 &= a_1(r) + a_2(r) \bmod d \\ &= [(b_1 + b_2) + 2\sum_{i=0}^{n-1}(u_{1,i} + u_{2,i})r^i]_d \end{aligned} \tag{7}$$

Donc le bruit u est additionnée.

Fonctionnement des opérations dans ce système ii

Multiplier des chiffrés

Après multiplication, on obtient

$$\begin{aligned} c_1 \times c_2 &= a_1(r) \times a_2(r) \bmod d \\ &= [(b_1 \cdot b_2) + b_1 \cdot U_2 + b_2 \cdot U_1 + U_1 U_2]_d \end{aligned} \tag{8}$$

où $U = 2 \sum_{i=0}^{n-1} u_i r^i$

En route vers le Fully Homomorphic

En route vers le FHE i

Nous avons jusque-là un système SWHE donc nous pouvons réaliser un certain nombre d'opérations (nombre qui dépendra des paramètres). L'idée est désormais de réaliser le Bootstrapping pour permettre de réinitialiser le nombre d'opérations possibles en recryptant le chiffré avec une clé chiffrée.

Seulement, notre fonction de décryptage est trop lourde pour pouvoir le faire actuellement, c'est pourquoi nous devons parler du **Squashing**.

Cela consiste tout simplement à réduire la consommation en opérations de notre décryptage tout en permettant de recrypter un chiffré.

En route vers le FHE ii

Squashing

Pour améliorer la fonction de décryptage, on va ajouter à la clé publique S ($\approx n$) entiers $x_1, \ldots, x_S \in \mathbb{Z}_d$ qui permettent de retrouver la clé privée w.

Ensuite, on va avoir un vecteur σ de taille S qui vaudra 0 partout sauf en $s(<< S)$ indices où il vaudra 1.

Et la somme des x_i pour les indices valant 0 dans σ vaudra w notre clé secrète

$$w = \sum_{i=1}^{S} \sigma_i x_i \bmod d$$

Squashing (2)

On peut désormais décrypter avec

$$dec(c) = \sum_{i=1}^{S} [\sigma_i y_i]_d \bmod 2$$

où $y_i = \langle c x_i \rangle_d$ (réduction modulo d dans l'intervalle [0, d[ )

Squashing (3)

Et comme on réduit à terme modulo 2, on arrive à terme à

$$dec(c) = \left(\bigoplus_{i=1}^{S} \sigma_i \langle y_i \rangle_2 \right) \oplus \left\langle \left\lceil \sum_{i=1}^{S} \sigma_i \frac{y_i}{d} \right\rfloor \right\rangle_2$$

Note : $\lceil \cdot \rfloor$ arrondit à l'entier le plus proche et $\bigoplus$ est le XOR soit l'addition modulo 2.

Squashing (4)

Pour continuer le squashing, il faut

- Transformer σ qui est de poids s en s vecteurs σ_i chacun de poids 1
- Approximer $\frac{y_i}{d}$ avec $p = log_2(s+1)$ bits de précision
- Implémenter un système d'addition avec les polynômes symétriques
- Crypter les σ_k en des polynômes de degré $log(S)$
- Implémenter une manière de retrouver les indices des x_k en multipliant des chiffrés ajoutés à la clé secrète

Squashing (5)

A ce stade, on peut effectuer le bootstrapping pour permettre de recrypter un chiffré et effectuer des opérations sur un chiffré à l'infini

Sécurité du cryptosystème i

Pour que un attaquant ayant seulement B_{pk} craque un chiffré $c = a \bmod B_{pk}$ pour un $a = 2u + be_1 \in \mathbb{R}^n$ où $r \in \{0,1\}^n$, il faut qu'il trouve a.

On sait que il existe un vecteur $v \in L$ tel que $v + a = c$, en effet le vecteur v est le point de L le plus proche de c, on a $\|v - c\|_2 < \sqrt{2(n+1)}$ et $\|v - c\|_\infty \leq 3$ ($\|v\|_\infty = \max_i |v_i|$).

Le problème est alors de trouver le vecteur $v \in L$ le plus proche de c on aura alors $a = c - v$.

Closest vector problem

Pour une base B d'un réseau et un vecteur $v \in \mathbb{R}^n$, on cherche le point du réseau le plus proche de v.

Ce problème est NP-difficile, ainsi que son approximation à un facteur constant.

Shortest vector problem

On note $\lambda(L)$ la longueur du plus petit vecteur du réseau L. Calculer $\lambda(L)$ est NP-difficile.

α-Bounded Distance Decoding Problem

Pour une base B d'un réseau et un vecteur $c \in \mathbb{R}^n$ tel que $d(c, L) < \alpha\lambda(L)$, on cherche le point du réseau le plus proche de v.

Ce problème est NP-difficile pour $\alpha > 1/\sqrt{2}$.

Force brute

Une autre attaque serait de tester tous les $a = 2u + 1$ où $u \in \{0, 1\}^n$ et trouver celui qui est chiffré en c. Le nombre de tests à effectuer est 2^{n+1}