Table des matières

Introduction à la cryptographie post-quantique

Léo Gagnon

Première partie

Motivation pour la nécéssité de nouveaux standards cryptographiques

À venir

Deuxième partie

Introduction à la cryptographie basée sur la théorie des codes

À venir

Troisième partie

Introduction à la cryptographie basée sur les réseaux

1 Préalables mathématiques

Nous introduirons d'abord quelques notions en lien avec les réseaux et nous poursuivrons avec la définition de plusieurs problème de complexité en lien avec ces réseaux.

1.1 Notation

On identifiera un vecteur par un caractère en gras (ex. $\mathbf{a}$) et une matrice par un caractère majuscule en gras (ex. $\mathbf{A}$).

Pour $x \in \mathbb{R}$, on notera $\lfloor x \rceil := \lfloor x + \frac{1}{2} \rfloor$ l'entier le plus proche de x.
On note ψ_σ la loi normale de moyenne 0 et écart-type σ.

1.2 Notions de réseau

Définition

Un réseau Λ de dimension n est un sous-groupe discret additif de $\mathbb{R}^n$. Autrement dit,

1. $\mathbf{0} \in \Lambda$ et $-x, x + y \in \Lambda \forall x, y \in \Lambda$
2. Chaque $x \in \Lambda$ possède un voisinage dans $\mathbb{R}^n$ dans lequel il est le seul élément de Λ

L'exemple typique de réseau est $\mathbb{Z}^n$.
Pour spécifier explicitement un réseau de dimension n, il suffit de donner n vecteurs indépendants $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ tel que $\Lambda = \{\sum_{i=1}^n z_i \mathbf{b}_i : z_i \in \mathbb{Z}\}$ (on appelle ces vecteurs la base de Λ).

Plus petit vecteur

Notons $\lambda_1(\Lambda)$ le plus petit vecteur non-nul appartenant à Λ.

Réseau dual

Le dual du réseau Λ, noté Λ^*, est le réseau donné par les points $y \in \mathbb{R}^n$ tel que $\langle x, y \rangle \in \mathbb{Z}$.

Loi normale sur réseau

On défini la loi normale discrète sur Λ avec paramètre r, noté par $D_{\Lambda,r}$, la distribution qui assigne une masse proportionnelle à $e^{-\pi||\mathbf{x}/r||^2}$ à chaque $\mathbf{x} \in \Lambda$. Ainsi, plus r est petit, plus la distribution est concentrée autour de $\mathbf{0}$. La norme d'un vecteur issu de $D_{\Lambda,r}$ est d'environ de $\sqrt{n}r$. Notons qu'on peut également voir $D_{\Lambda,r}$ comme la loi normale de moyenne 0 et variance r discrétisé aux points de Λ.

1.3 Problèmes calculatoires

Définition 1 (Learning with errors, LWE)**.** *Soient $n \geq 1$, $q \geq 2$, $\mathbf{s} \in \mathbb{Z}_q^n$ et une distribution d'erreur χ fixé sur $\mathbb{Z}_q$. Pour les applications cryptographiques, $\chi = \psi_{\alpha q}$ pour $\alpha \in (0, 1)$ (α est appelé le terme d'erreur). Notons $A_{\mathbf{s},\chi}$ la distribution sur $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtenue en choisissant $\mathbf{a} \in \mathbb{Z}_q^n$ aléatoirement de façon uniforme et e selon χ et en retournant $(\mathbf{a}, (\langle \mathbf{a}, \mathbf{s} \rangle + e) \mod q)$. On dit qu'un algorithme résout LWE avec ces paramètres si il peut déterminer $\mathbf{s}$ avec un nombre arbitraire d'échantillions indépendents de $A_{\mathbf{s},\chi}$. Il existe une version décisionelle de LWE (que l'on nomera dec-LWE), soit distinguer $A_{\mathbf{s},\chi}$ de la distribution uniforme sur $\mathbb{Z}_q^n \times \mathbb{Z}_q$. On montrera la réduction de la version de recherche à la version décisionelle.*
Remarquons qu'une façon alternative de définir le problème LWE est la suivante : Soient $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{e} \in \chi^m$ et $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$. Étant donné $\mathbf{As} + \mathbf{e}$ trouver $\mathbf{s}$. Il est facile de voir que cela équivaut à résoudre LWE avec m échantillions de $A_{\mathbf{s},\chi}$.

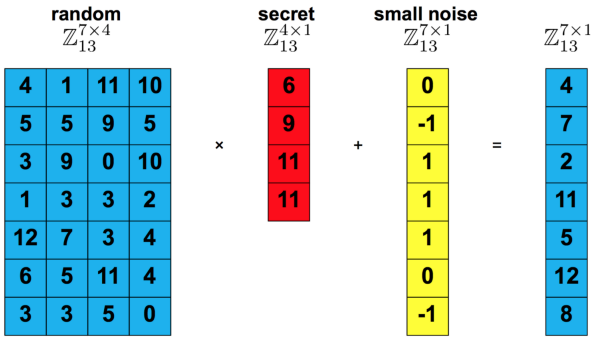

FIGURE 1: Exemple d'une instance LWE avec $n = 4, q = 13$ et $m = 7$

Définition 2 (Shorter integer solution, SIS)**.** *Soit* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *une matrice* $n \times m$ *constituée de* m *vecteurs aléatoires choisis uniformément dans* $\mathbb{Z}_q^n$*. L'objectif est de trouver* $\mathbf{x} \in \mathbb{Z}^m$ *non-nul tel que* $||\mathbf{x}|| \leq \beta$ *et* $f_{\mathbf{A}} := \mathbf{Ax} = \mathbf{0}$*.*

Définition 3 (Bounded distance decoding, BDD)**.** *Soit un réseaux* Λ *et un point* $\mathbf{x}$ *à une distance maximale* $d > 0$ *de* Λ *(le plus court vecteur reliant* $\mathbf{x}$ *à un point* $\mathbf{y} \in \Lambda$ *a une norme inférieure à* d*). L'objectif est de trouver le point* $\mathbf{y} \in \Lambda$ *le plus près (distance euclidienne) de* $\mathbf{x}$*. Remarquons la réponse est unique si et seulement si* $d < \lambda_1(\Lambda)/2$*.*

Définition 4 (Shortest vector problem, SVP & GapSVP)**.** *Soit un réseau* Λ*, SVP demande de trouver* $\lambda_1(\Lambda)$ *tant dis que GapSVP demande de distinguer entre les cas* $\lambda_1(\Lambda) \leq d$ *et* $\lambda_1(\Lambda) > d\gamma$*. GapSVP est donc un version approximative de SVP où* γ *est le facteur d'approximation.*

1.3.1 Complexité de (Gap)SVP

Nous verrons plus loin que la sécurité de la cryptographie à base de réseaux est basée sur la difficulté de GapSVP. En particulier, la propriété post-quantique vient du fait qu'à ce jour aucun algorithmes quantique n'est capable de résoudre efficacement GapSVP. Il est donc important de bien comprendre la difficultée de GapSVP.

L'état de nos connaissances est le suivant : GapSVP est NP-difficile avec des facteurs d'approximation $\gamma = o(\sqrt{n})$ mais probablement pas pour $\gamma \geq \sqrt{n}$ (voir [**ar04**] pour plus de détails). Malheureusement, à ce jour, les applications cryptographiques dépendent de facteurs d'approximation $\gamma \geq n$ donc ne bénificient pas de difficultée d'un problème NP-difficile.

Cependant, (GapSVP) a été beaucoup étudié et aucun algorithme (autant quantique que classique) n'est capable de le résoudre efficacement sauf avec des facteurs d'approximations exponentiels.

1.3.2 Lien avec la théorie des codes

Il est possible de voir LWE comme une sorte de généralisation d'un problème fondamental en théorie des codes : le décodage de codes aléatoires.
Rappellons brièvement ce problème.

Définition 5 (Random linear code decoding, RLCD)**.** *Soit une matrice aléatoire* $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ *servant de matrice génératrice à un code linéaire. Étant donné un mot encodé entaché d'une erreur* $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ *où* $\mathbf{x} \in \mathbb{Z}_2^n$ *et* $\mathbf{e}$ *est choisi aléatoirement dans* $\mathbb{Z}_2^m$ *tel que le décodage est possible, trouver* $\mathbf{x}$*.*

On peut remarquer que résoudre RLCD revient à résoudre LWE ($q = 2$) avec m échantillions $(\mathbf{a}_i, b = \mathbf{a}_i \cdot \mathbf{x} + e_i)$. Où $\mathbf{a}_i$ est à le ième rangée de A, $\mathbf{x}$ est le mot encodé et e_i est l'erreur d'encodage.

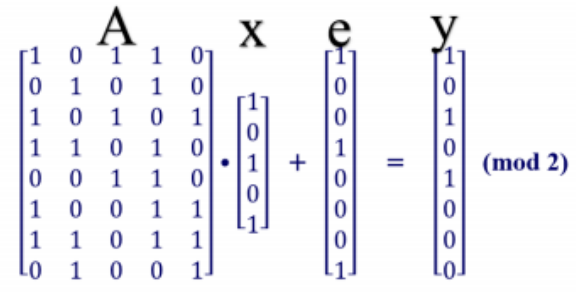

FIGURE 2: Instance de RLCD = Instance de LWE avec $q = 2$

2 Problème LWE

Dans cette section, nous discuterons de la difficultée du problème LWE en exposant différentes réductions et en expliquant pour quelles conditions les hypothèse de difficulté calculatoire tiennent.

2.1 Résultats de complexité

2.1.1 Réduction classique

Théorème 1 ([Reg05])**.** *Soient* $n \geq 1, q \geq 2$ *et* $\alpha \in (0, 1)$*. Supposons que l'on possède un oracle pour LWE avec comme paramètre* q *et* α*. Alors, pour un réseau* Λ*, avec assez d'échantillions de* $D_{\Lambda^*, r}$*, on peut résoudre BDD en temps polynomial avec* $d \leq \frac{\alpha q}{\sqrt{2}r}$*.*

Idée de la preuve ($\Lambda = \mathbb{Z}^n$ est utilisé pour l'intuition bien que l'argument général soit semblable)**.** *Soit un point* $\mathbf{x} \in \mathbb{R}^n$ *et* $\mathbf{v}$ *le point de* Λ *le plus proche de* $\mathbf{x}$*. Premièrement, remarquons que si nous pouvons générer des échantillions de LWE avec comme secret* $\mathbf{s} = \mathbf{v}$ *mod* q*, alors un oracle pour LWE permet de retrouver* $\mathbf{v}$ *(résoudre BDD) de la manière suivante :* $\mathbf{s}_0 = \mathbf{v}$ *mod* q *est égal au premier bit de* $\mathbf{v}$ *en base* q*,* $\mathbf{s}_1 = \frac{(\mathbf{v}-\mathbf{s})}{q}$ *mod* q *est égal au deuxième et ainsi de suite.* $\mathbf{v}$ *est donc égal*

à $(\dots \mathbf{s}_3\mathbf{s}_2\mathbf{s}_1\mathbf{s}_0)_q$*. Afin de générer de tels échantillions, on choisi* $\mathbf{y} \in D_{\mathbb{Z}^n,r}$ *(*$\mathbb{Z}^n$ *est son propre dual) et on retourne la paire*

$$(\mathbf{a} = \mathbf{y} \ mod \ q, \lfloor \langle \mathbf{y}, \mathbf{x} \rangle \rceil mod \ q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

On peut montrer que cette paire suit bien la distribution LWE appropriée avec l'erreur suivant une loi normale. L'idée générale vient du fait que puisque $\mathbf{x} = \mathbf{v} + \mathbf{e}$ *alors* $\langle \mathbf{y}, \mathbf{x} \rangle \ mod \ q = \langle \mathbf{y}, \mathbf{v} \rangle \ mod \ q + e = \langle \mathbf{a}, \mathbf{s} \rangle + e$*. La restriction sur la distance est due à des technicalités.*

Déjà, cela nous donne une bonne idée de la difficultée de LWE puisque le meilleur algorithme que nous connaissons pour BDD fonctionne en temps exponentiel (pour $\alpha q = \Omega(\sqrt{n})$). Cependant, il semble qu'une réduction au problème plus standard GapSVP soit souhaité puique la complexité de ce problème est mieux comprise. Cette réduction est plus complexe donc on ne fait que l'énoncer et référer à [LM09] pour les détails.

Théorème 2 ([LM09])**.** *Supposons que l'on possède un oracle pour BDD avec comme paramètre* $d \leq \frac{\lambda_1(\Lambda)}{poly(n)}$*, alors on peux résoudre GapSVP pour* $\gamma = poly(n)$

En combinant les théorèmes 1 et 2 on peut voir que tant que $\frac{\alpha q}{\sqrt{2}r} \geq \frac{\lambda_1(\Lambda)}{poly(n)}$, une solution pour LWE implique une solution pour GapSVP. Cependant, on doit être capable de générer des échantillions de $D_{\Lambda^*,r}$. [GPV08] a montré que ceci est possible si l'on possède une base pour Λ ayant des vecteurs de longeur r. Or, il est possible de produire efficacement une telle base de longeur $\frac{2^n}{\lambda_1(\Lambda)}$. En remplacant $r = \frac{2^n}{\lambda_1(\Lambda)}$ dans la formule voit que q doit être $2^{O(n)}$.
Une solution à LWE pour un moduli exponentiel est donc équivalent à une solution de GapSVP pour $\gamma = poly(n)$.

Théorème 3 ([Bra+13])**.** *Supposons que l'on possède un oracle pour LWE avec moduli* $p = poly(n)$ *et dimension* n*, alors on peut résoudre LWE avec un moduli* $q > poly(n)$ *et dimension* $\sqrt{n}$*.*

Avec le résultat du Théorème 3, on conclu que LWE (pour $q = poly(n)$ et dimension n) est aussi difficile que GapSVP (de dimension $\sqrt{n}$).

2.1.2 Réduction quantique

Puisque cette réduction dépasse le cadre de ce papier donc nous contenterons l'énoncer et référer à [Reg05] pour les détails.

Théorème 4 ([Reg05])**.** *Soient* $q \leq poly(n)$ *et* $\alpha \in (0, 1)$ *tels que* $\alpha q > 2\sqrt{n}$*. Supposons que l'on possède un oracle pour LWE (moduli* q*, dimension* n *et* $\chi = \psi_\alpha$*) et un nombre polynomial d'échantillions, alors il existe un algorithme quantique efficace pour résoudre GapSVP (avec* $\gamma = \tilde{O}(n/\alpha)$ *et dimension* n*)*

Notons que cette réduction est "meilleure" que la précédente car elle conserve la dimension (c'est un problème ouvert de trouver une réduction classique qui conserve la dimension). Aussi, c'est elle qui est utilisée dans les preuves de sécurité des systèmes cryptographiques car elle offre une relation simple entre le terme d'erreur α et le facteur d'approximation γ pour GapSVP.

2.2 Propriétés utiles

2.2.1 Cas moyen vers pire cas

Théorème 5 ([Reg05]). *Soient $n \geq 1, q \geq 2$ et χ une distribution sur $\mathbb{Z}_q$. Supposons que l'on ai accès à un algorithme W qui distingue $A_{\mathbf{s},\chi}$ de U pour une portion non-négligeable de tout les $\mathbf{s}$. Alors il existe un algorithme efficace W' qui distingue $A_{\mathbf{s},\chi}$ de U pour tout $\mathbf{s}$ avec une probabilité exponentillement près de 1.*

Idée de la preuve. *Soit $f_t : \mathbb{Z}_q^n \times \mathbb{Z}_q \to \mathbb{Z}_q^n \times \mathbb{Z}_q$ tel que :*

$$f_t(\mathbf{a}, b) = (\mathbf{a}, b + \langle \mathbf{a}, \mathbf{t} \rangle)$$

On peut voir que cette f_t transforme la distribution $A_{\mathbf{s},\chi}$ en $A_{\mathbf{s}+\mathbf{t},\chi}$ et la distribution uniforme U en elle-même. L'algorithme fonctionne donc de la manière suivante (pour un nombre polynomial d'itération) : appliquer f_t à la distribution (pour $t \in \mathbb{Z}_q^n$ choisi aléatoirement), appliquer W à la distribution résultante et à U, répondre OUI si W distingue les deux, sinon recommencer. Refuser à la fin.
On voit qu'avec une très grande probabilité on va éventuellement tomber sur un $A_{\mathbf{s}+\mathbf{t},\chi}$ que W est capable de distinguer de U.

2.2.2 Décision vers recherche

Théorème 6 ([Reg05]). *Soient $n \geq 1$, $2 \leq q \leq poly(n)$ un nombre premier et χ une distribution d'erreur sur $\mathbb{Z}_q$. Supposons que l'on ai accès à un oracle pour dec-LWE, alors il existe un algorithme W qui retrouve $\mathbf{s}$ à partir de $A_{\mathbf{s},\chi}$ avec probabilité exponentiellement près de 1.*

Idée de la preuve. *La procédure suivante trouve la première coordonnée de $\mathbf{s}$ avec un oracle pour dec-LWE. Soit $f_{t,k} : \mathbb{Z}_q^n \times \mathbb{Z}_q \to \mathbb{Z}_q^n \times \mathbb{Z}_q$ tel que :*

$$f_{t,k}(\mathbf{a}, b) = (\mathbf{a} + (r, 0, \ldots, 0), \mathbf{b} + r \cdot k)$$

où $r, k \in \mathbb{Z}_q$. Il est facile de voir que $f_{t,k}$ conserve $A_{\mathbf{s},\chi}$ seulement si $k = s_1$ (on peut montrer que si $k \neq s_1$ alors $f_{t,k}$ transforme $A_{\mathbf{s},\chi}$ en U). Ainsi, on a qu'à appliquer $f_{t,k}$ sur $(\mathbf{a}, b)$ pour tout $k \in \mathbb{Z}_q$ avec r aléatoire sur $\mathbb{Z}_q$ et le k pour lequel l'oracle distingue $f_{t,k}(\mathbf{a}, b)$ de U sera égal à s_1.

2.2.3 Indépendance du nombre d'échantillion

Remarquons que d'après la preuve du Théorème 1, une instance du problème LWE est elle même une instance d'un problème BDD. Ainsi, étant donné un $m = poly(n)$ échantillions d'une instance LWE, on peut trouver le point $\mathbf{x}$ du problème BDD sous-jacent et ensuite générer un nombre arbitraire d'échantillion pour le problème LWE (avec la procédure expliquée dans la preuve). Notons que peu importe la distribution de l'erreur des échantillions initiaux, ceux crées avec cette méthode aurons tous une erreur normale. Cela explique (intuitivement) pourquoi la difficultée du problème de dépends pas du nombre d'échantillions.

2.3 Exemple d'encryptio à clée privée

Définissons un schéma d'encryption à clé privée pour un espace de message $\mathcal{M} := \{0,1\}$ basé sur LWE. Notons qu'il existe de nombreuses autres applications cryptographiques (cryptographie à clée publique, chiffrement homomorphe, ...).

$Gen(1^\lambda)$: Choissir la clée privée

$$k := \mathbf{s} \in \mathbb{Z}_q^n$$

aléatoirement de façon uniforme.

$Enc(k, m)$: Générer le cryptogramme

$$c := (\mathbf{a}, b) := (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e + m\lfloor q/2 \rceil \ mod\ q)$$

où $\mathbf{a} \in \mathbb{Z}_q^n$ et $e \in \chi$ son choisis aléatoirement de façon uniforme.

$Dec(k, c = (\mathbf{a}, b))$: Retourner $m = 0$ si

$$|b - \langle \mathbf{a}, \mathbf{s} \rangle \ mod\ q| = |e + m\lfloor q/2 \rceil \ mod\ q| < q/4$$

et 1 sinon.

Théorème 7. *La construction ci-haut (avec quelques détails supplémentaires) est correcte si le support de la distribution d'erreur $Supp(\chi) \in (-q/4, q/4)$ et est résistante aux attaques CPA sous l'hypothèse que dec-LWE est difficile pour un nombre polynomial d'échantillions ($A_{\mathbf{s},\chi}$ est indistingable de la distribution uniforme sur $\mathbb{Z}_q^n \times \mathbb{Z}_q$).*

Notons que ce cryptosystème, ainsi que dans tout ceux basés sur LWE, on doit ajuster les paramètres pour la difficultée soit basée sur GapSVP.

2.4 Source de l'inefficacité et solution

Comme nous l'avons vu, la version classique du problème défini sur des réseaux est très inefficace. Par exemple, dans la construction à clée privée, il faut un cryptogramme de longeur $O(n^2)$ pour encrypter $O(n)$ bits d'information. De plus, puisque les systèmes cryptographiques issus de LWE utilisent des grands vecteurs, ils requierent beaucoup d'espace mémoire et doivent faire beaucoup de calcul peu optimisés (multiplication de vecteurs).
On vera comment améliorer l'efficacité des systèmes cryptographiques basé sur les réseaux dans la section 4.

3 Problème SIS

Ce problème, introduit par [MR07], est très lié à LWE et ses applications cryptographiques sont davantage sous la forme d'outils (fonction à sens unique, fonction de hachage résistante au collisions, schéma d'identification, ...).

3.1 Résultat de complexité

Tout comme pour LWE, la difficultée de SIS est basée sur le problème classique de réseau GapSVP. Dû à sa nature très intuitive, on se contentera de citer le théorème (simplifié) et d'expliquer brievement l'intuition. Les détails se trouvent dans [**mr07**]

Théorème 8 ([MR07])**.** *Soit $q \geq \beta(n)$ pour un certain polynôme $\beta(n)$. Alors GapSVP se réduit au cas moyen de SIS (paramètre q).*

L'intuition est la suivante : il est facile de voir que l'ensemble $\Lambda = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{Az} = 0 \ mod \ q\}$ forme un réseau, donc résoudre SIS avec la matrice $\mathbf{A}$ revient à trouver le plus petit vecteur de Λ (SVP/GapSVP). Pour résoudre GapSVP avec Λ arbitraire, on a qu'à déterminer pour quel $\mathbf{A'}$ on a $\Lambda = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A'z} = 0 \ mod \ q\}$ et ensuite résoudre SIS avec $\mathbf{A'}$.

3.2 Exemple de fonction de hachage résistante au colisions

Définissons une famille de fonction de hachage résistante au collisions basé sur le problème SIS. Soit $h_{\mathbf{A}} : \{0, \ldots, B\}^m \to \mathbb{Z}_q$ tel que :

$$h_{\mathbf{A}}(\mathbf{e}) = \mathbf{Ae} \ mod \ q$$

où $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. Une collision dans $h_{\mathbf{A}}$ donne $\mathbf{e}, \mathbf{e'} \in \{0, \ldots, B\}^m$ tel que $\mathbf{A}(\mathbf{e} - \mathbf{e'}) = 0 \ mod \ q$. On a donc que $\mathbf{e} - \mathbf{e'} \in \{-B, \ldots, B\}^m$ est une solution pour SIS avec paramètres n, m, q et B. Ainsi, la famille de fonction de hachage $\mathcal{H}$ contenant les $h_{\mathbf{A}}$ est résistante au collision avec l'hypothèse que SIS est difficile.

Notons que l'utilisation d'une grande matrice $\mathbf{A}$ rend cette fonction de hachage très peu efficace (comme pour LWE). Il est à noter que l'efficacité de SIS peut être amélioré de la même manière que celle de LWE (comme on le vera dans la prochaine section).

4 ring-LWE et ring-SIS

Dans cette section, nous expliquerons comment structurer la matrice $\mathbf{A}$ (définitions 1 et 2) et nous redéfinirons les problèmes SIS et LWE à l'aide de cette observation. La problèmes résultant pourrons ensuite être utilisés pour rendre plus efficaces les constructions vues dans les sections précédentes.

Il est important que comprendre que l'exemple de structure qu'on donne n'est pas le seul et qu'il est possible de généraliser tout ce qu'on vera dans cette section à l'aide d'objets mathématiques plus abstraits. Nous éviterons cependant le formalisme mathématique afin de favoriser la compréhension.

4.1 Réseaux idéaux

La difficulté des problèmes ring-SIS et ring-LWE, tout comme celle de SIS et LWE, est basée sur la difficulté de trouver des vecteurs courts dans un réseau. La seule différence est que pour les versions

sur anneaux, on se restreint à un certain type de réseau qu'on appelle *réseau idéal*. Nous éviterons les détails techniques en lien avec ces objets mathématiques car cela dépasse le cadre de ce document.

Bien évidement, ces problèmes sont d'intérêt seulement si la recherche d'un cours vecteur dans ces réseaux particuliers reste difficile. Au meilleur de nos connaissances, il n'existe pas d'algorithmes (même quantiques) qui peuvent approximer efficacement SVP sur des réseaux idéaux (sauf pour de très grandes valeurs de γ, ce qui ne pose pas problème). Ainsi, il est très probable que trouver des vecteurs courts d'un réseau idéal soit aussi difficile que pour un réseau arbitraire.

Les détails à propos de la difficultée de ring-SIS et ring-LWE peuvent être trouvés dans [LPR10] et [**lpr13**].

4.2 Matrice de décalage anti-cyclique et anneau $\mathbb{Z}[x]/\langle x^n + 1\rangle$

Nous présentons ici une structure qu'on peut donner à la matrice $\mathbf{A}$ afin de pouvoir l'exprimer avec beaucoup moins d'information.

Définissons la matrice de décalage anti-cyclique d'un vecteur $\mathbf{a}$ de la façons suivante :

$$Rot(\mathbf{a}) := \begin{bmatrix} a_1 & -a_n & -a_{n-1} & \dots & -a_2 \\ a_2 & a_1 & -a_n & \dots & -a_3 \\ a_3 & a_2 & a_1 & \dots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & -a_n \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{bmatrix} \in \mathbb{Z}^{n\times n}$$

Ainsi, $Rot(\mathbf{a})$ est la matrice ayant pour colonnes le vecteur $\mathbf{a}$ ainsi que toutes ses permutations anti-cycliques. Remarquons que on peut également écrire $Rot(a) := (\mathbf{a}, X\mathbf{a}, X^2\mathbf{a}, \dots, X^{n-1}\mathbf{a})$ où

$$X := \begin{bmatrix} 0 & 0 & \dots & 0 & -1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \in \{0,1\}^{n\times n}$$

Considérons maintenant $\tilde{R} := \{Rot(\mathbf{a}) : \mathbf{a} \in \mathbb{Z}^n\} \subset \mathbb{Z}^{n\times n}$ l'ensemble de tout les matrices anti-cycliques entières. Remarquons que $\tilde{R}$ est un réseau en $n \times n$ dimension avec comme base $I_n, X, X^2, \dots, X^{n-1}$ (on peut écrire $Rot(\mathbf{a}) = a_1 I_n + a_2 X + \dots + a_n X^{n-1}$). Il est également facile de voir que $\tilde{R}$ forme un anneau. Plus interressant encore, $\tilde{R}$ est isomorphique à l'anneau de polynôme $R := \mathbb{Z}[x]/\langle x^n + 1\rangle$ (l'ensemble des classes de résidu des polynômes $\mathbb{Z}[x]$ modulo $(x^n + 1)$) car on peut tout simplement appliquer $X \mapsto x$ (ou l'inverse) pour passer d'un anneau à l'autre (on peut vérifier que cette relation

est bien un isomorphisme).
Ceci explique donc comment redéfinir SIS sur un anneau de polynôme augmente l'efficacité des applications : au lieu d'utiliser une matrice $\mathbf{A} \in \mathbb{Z}^{n \times n}$ on peut utiliser $Rot(\mathbf{a}) \in \tilde{R}$ ou de façon équivalente $a \in R$ (a est le polynôme de degré $n-1$ ayant comme coefficients les éléments de $\mathbf{a}$). Notons aussi que les opérations $(+, \times)$ sur R sont très efficaces (grâce à la transformation de Fourier rapide).

4.3 Reformulation de SIS

Définissons maintenant le problème SIS sur R :

Définition 6 (ring-SIS)**.** *Soit l'anneau $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$, un moduli $q \geq 2$ et un entier $l \geq 1$. Étant donné $a_1, \ldots, a_l \in R_q$ tous choisis aléatoirement de façon uniforme, le but est de trouver $e_1, \ldots, e_l \in R$ non tous-nuls tels que $a_1 e_1 + \ldots + a_l e_l = 0$ mod qR (mod qR signifie qu'on réduit le polynôme modulo $(x^n + 1)$ et ensuite les coefficients modulo q) et $0 < ||(\rho(e_1), \rho(e_2), ..., \rho(e_l))|| \leq \beta$ (ρ transforme un polynôme en son vecteur coordonnées).*

Notons que cette définition, ainsi que tout les théorèmes reliés, peuvent être généralisés à d'autres anneaux et d'autres types de normes mais par souci de simplicité nous nous concentrerons sur celle-ci.

4.4 Fonction de hachage efficace

Comme nous avons vu plus haut, on peut maintenant exprimer une matrice $n \times n$ avec un polynôme de $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ (ou de façon équivalente son vecteur de coordonées). Ainsi, on exprimera la matrice $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ comme $l = m/n$ éléments de R_q et la fonction de hachage deviendra $h_{a_1,\ldots,a_l}(e_1, \ldots, e_l) = a_1 e_1 + \ldots + a_l e_l$ mod qR. On peut voir qu'une collision dans cette fonction résulte en une solution à ring-SIS. En pratique, le message de longeur m est encodé dans les coefficients des polynômes. Sans rentrer dans les détails, cette fonction de hachage nécéssite une clée de longeur $O(n \log n)$ et fonctionne en temps quaisi-linéaire, ce qui la rend utilisable en pratique.

4.5 Reformulation de LWE

Puisqu'on sait qu'on peut exprimer une matrice $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ avec un polynôme $a \in R_q$, alors la reformulation de LWE revient à "compresser" n échantillons LWE classiques dans un échantillon ring-LWE (si on considère la formulation alternative de la définition 1). Ainsi, chaque échantillon cache n bit pseudo-aléatoires au lieu de un.

Définition 7 (ring-LWE)**.** *Soit n une puissance de 2 et q un nombre premier satisfaisant $q = 1$ mod $2n$. Notons $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ l'anneau contenant tout les polynômes sur $\mathbb{Z}_q$ modulo $(x^n + 1)$. Étant donné des échantillions $(a, b = a \cdot s + e) \in R_q \times R_q$, où $s \in R_q$ est le secret, $a \in R_q$ est choisi uniformément et e est un terme d'erreur choisi selon χ, le but est de retrouver s. La version décisionnelle demande de distinguer $(a, b = a \cdot s + e)$ de (a, b) tiré d'une distribution uniforme sur $R_q \times R_q$.*

Notons ici que ring-LWE possède les même propriétés que LWE : cas moyen difficile, équivalence recherche-décision et indépendance du nombre d'échantillons. Les réductions sont analogues (à part des détails techniques).

4.6 Encryption à clée privée efficace

Définissons un schéma d'encryption à clé privée pour un espace de message $\mathcal{M} := R_2$ (éléments de R avec coefficients 0,1) basé sur ring-LWE. Remarquons que l'espace de message est n fois plus grand que dans la construction avec LWE.

$Gen(1^\lambda)$: Choissir la clée privée

$$k := s \in R_q$$

aléatoirement de façon uniforme.

$Enc(k, m)$: Générer le cryptogramme

$$c := (\mathbf{a}, b) := (a, a \cdot s + e + m\lfloor q/2 \rceil \ mod \ qR)$$

où $a \in R_q$ et $e \in \chi$ son choisis aléatoirement de façon uniforme.

$Dec(k, c = (a, b))$: Calculer

$$\hat{m} = b - a \cdot s \ mod \ qR = m\lfloor q/2 \rceil + e$$

Arrondir chaque coefficient de $\hat{m}$ à 0 ou $q/2$. Interpréter 0 comme 0 et $q/2$ comme 1.

Théorème 9. *La construction ci-haut (avec quelques détails supplémentaires) est correcte si le support de la distribution d'erreur $Supp(\chi) \in (-q/4, q/4)$ et est résistante aux attaques CPA sous l'hypothèse que la version décisionelle ring-LWE est difficile pour un nombre polynomial d'échantillions (distinguer $(\mathbf{a}, \mathbf{b} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e})$ de $(\mathbf{a}, \mathbf{b})$ tiré d'une distribution uniforme sur $R_q \times R_q$).*

Cette nouvelle construction permet d'obtenir un facteur d'expension du cryptogramme de $O(1)$ (contre $O(n)$ pour LWE) et fonctionne plus rapidement grace à l'efficacité des opérations sur R.

Quatrième partie

Construction cryptographiques post-quantiques

5 Fonctions pseudo-aléatoires (détails ?)

Premièrement, remarquons la nature aléatoire de LWE (erreur échantillonnée de χ) fait en sorte qu'elle ne se porte pas bien à la construction de fonctions pseudo-aléatoires. Définissons une variante de LWE qui résout ce problème.

Définition 8 (Learning with Rounding, LWR)**.** *Soient $n \geq 1$, $q > p$ et $\mathbf{s} \in \mathbb{Z}_q^n$. Notons $A_{\mathbf{s}}$ la distribution sur $\mathbb{Z}_q^n \times \mathbb{Z}_p$ obtenue en choisissant $\mathbf{a} \in \mathbb{Z}_q^n$ aléatoirement de façon uniforme et en retournant $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p)$ (où $\lfloor \cdot \rceil_p$ est la valeur arrondie mod p). On dit qu'un algorithme résout LWE avec ces paramètres si il peut déterminer $\mathbf{s}$ avec un nombre arbitraire d'échantillions indépendents de $A_{\mathbf{s}}$.*

[**akp13**] a montré une réduction de LWE à LWR pour $q \geq poly(n)$ (détails ?).

LWR est donc analogue à LWE avec la seule différence que l'erreur est "déterministe" est non tirée aléatoirement d'une distribution d'erreur. [**bpr**] a montré que l'on peut construire une famille de fonctions pseudo-aléatoires basée sur LWR en utilisant une construction semblablable à celle de [**nrr00**] (détails ?). Plus précisement, la famille définie dans [**bpr**] est la suivante :

$$F_{\mathbf{a},\{\mathbf{S}_i\}}(x) = \lfloor \mathbf{a}^t \cdot \prod_{i=1}^{l} \mathbf{S}_i^{x_i} \rceil_p$$

où la clée secrète est $\mathbf{a} \in \mathbb{Z}_q^n$ et l matrices $\mathbb{S}_i \in \mathbb{Z}^{n \times n}$ provenant d'une distribution gaussienne.

Cette fonction est prouvée sécure seulement pour $q \geq n^l$ mais les auteurs de la preuve croient que cette restriction est dû aux techniques de preuves utilisés (qu'elle n'est pas intrinsèque) (détails ?).

Notons que [**bbl14**] a proposé une implémentation pratique efficace (SPING) de la version sur anneau de cette fonction (détails ?). Elle est environ 4 fois plus lente que AES en mode CRT.

6 Fonctions de hachage efficace SWIFFT

Nous présentons ici la famille de fonction de hachage résistante au collisions SWIFFT issu de [**swifft**]. L'idée générale de la construction est essentiellement la même que la fonction de hachage canonique de ring-SIS (en plus efficace). La sécurité de SWIFFT est basé sur ring-SIS (avec $R = \mathbb{Z}[x]/\langle x^n + 1 \rangle$) et son efficacité est relativement bonne (pour une aussi bonne garantie de sécurité). Un fonction de la famille SWIFFT est définie par les paramètres m, n, q et m vecteurs $\mathbf{a}_i \in \mathbb{Z}_q^n$ correspondant aux coefficients de Fourier de polynômes choisies aléatoirement dans R.

1. Prendre en entré un message binaire M de longeur $m \cdot n$ interpreté comme une m vecteurs $\mathbf{x}_i \in \{0,1\}^n$.
2. Multiplier par coordonnées chaque $\mathbf{x}_i$ par $(\omega^0, \ldots, \omega^{n-1})$ (pour un $\omega \in \mathbb{C}$ fixé en fonction de m, n et q). Cela a pour but d'accomplir la diffusion.
3. Convertir chaque $\mathbf{x}_i$ en un polynôme $f_i \in R_2$.
4. Calculer les coefficients de Fourier $\mathbf{y}_i \in \mathbb{Z}_q^n$ de chaque f_i en utilisant la TFR.
5. Calculer $z_i = \sum_{j=1}^{m} a_{j,i} \cdot y_{j,i}$ (multiplication par coordonnées). Cela revient à calculer les produits de polynômes.

6. Retourner le vecteur $(z_1, \ldots, z_n) \in \mathbb{Z}_q^n$. Notons que ce vecteur identifie un unique polynôme de R.

Concrètement, avec les paramètres $n = 64, m = 16, p = 257$ et $(\mathbf{a}_1, \ldots, \mathbf{a}_m)$ choisi aléatoirement, on obtient un input de 1024 bits, un output de 66 bits et une image de grandeur 2^{512}.
Le principal avantage de SWIFFT est sa preuve de sécurité (basé sur ring-SIS donc probablement résistant à l'ordinateur quantique) et son temps d'exécution comparable à celui de SHA-256.
Son principal désavantage est le fait qu'elle ne constitue pas une fonction pseudo-aléatoire et qu'elle est tout de même moins rapide que des fonctions de hachage sans preuve de sécurité.

7 Encryption CPA-sécure avec ring-LWE

Décrivons brièvement le schéma "cannonique" d'encryption à clé publique (CPA-sécure) basé sur ring-LWE. Notons $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ avec n une puissance de 2.

— La clée secrète est $sk := s \leftarrow \chi$ où χ est une distribution d'erreur.

— La clée publique est un échantillon ring-LWE

$$pk := (a, b \approx s \cdot a \; mod \; Rq) \in R_q \times R_q$$

pour $a \in R_q$ et une erreur tirée de χ.

— Pour encrypter $m \in R_2$ (correspondant à un message binaire de longeur n), on génère le cryptogramme suivant :

$$c = (u \approx a \cdot r, v \approx b \cdot r + m \cdot \lfloor \frac{q}{2} \rceil) \in R_q \times R_q$$

— Pour décrypter $c = (u, v)$, on calcule

$$\hat{m} = v - s \cdot u \approx m \cdot \lfloor \frac{q}{2} \rceil + b \cdot r - s \cdot a \cdot r \approx m \cdot \lfloor \frac{q}{2} \rceil + s \cdot a \cdot r - s \cdot a \cdot r \approx m \cdot \lfloor \frac{q}{2} \rceil$$

Avec q et χ choisis adéquatement, on peut retrouver m en arrondissant les coefficients $\hat{m}$ soit à 0 soit à $\frac{q}{2}$ et en remplacent les $\frac{q}{2}$ à 1.

8 Fonctions à trappe avec SIS et LWE

Il existe deux façons (naturelles) de créer des fonctions à trappes avec les problèmes de réseaux. Essentiellement, ces techniques se basent sur le fait que les problèmes de réseaux (comme SIS et LWE) sont faciles si l'on possède un base particulière (qui sert de trappe). Nous donnerons plus bas l'idée générale de ces deux méthodes.

8.1 Bases courtes de [gpv08]

8.2 Bases "gadget" de [mp12]

À venir ?

9 Fonction à trappe lossy de [pw08]

9.1 Définition

Une famille de fonctions à trappe lossy (LTDF) est définie par deux paramètres : n représente la taille de l'entré et k représente le nombre de bit dont la fonction lossy perd les informations (en anglais la "lossiness" de la fonction). Notons $r = n - k$ le "residual leakage" d'une LTDF, cette valeur correspond à la quantité de bits de l'image que la fonction lossy laisse s'échapper. Notons que plus r est petit, meilleur est la LTDF.

Formellement, une collection de (n, k)-LTDF est donné par un tuple d'algorithmes polynomiaux $(S_{ltdf}, F_{ltdf}, F_{ltdf}^{-1})$ ayant les propriétés suivantes :

- $S_{ltdf}(1^\lambda, 1) = S_{inj}(1^\lambda)$ retourne (s, t) où s est l'index d'une fonction et t sa trappe. On a alors que $F_{ltdf}(s, \cdot)$ calcule une fonction $f_s(\cdot)$ injective sur $\{0,1\}^n$ et $F_{ltdf}^{-1}(s, \cdot)$ calcule $f_s^{-1}(\cdot)$.
- $S_{ltdf}(1^\lambda, 0) = S_{loss}(1^\lambda)$ retourne $(s, \perp)$ où s est l'index d'une fonction $f_s(\cdot)$ sur $\{0,1\}^n$ ayant une image de taille maximale $2^r = 2^{n-k}$
- Les premiers résultats de $S_{inj}(1^\lambda)$ et $S_{loss}(1^\lambda)$ sont indistinguables par un adversaire efficace.

9.2 Idée générale (à reformuler ?)

Cette LTDF consiste à multiplier un message $\mathbf{x} \in \mathbb{Z}_2^n$ par l'encryption $\mathbf{C}$ d'une matrice $\mathbf{M}$ (par un cryptosystème CPA-sécure issu de LWE). Ce cryptosystème doit possèder les propriétés suivantes : homomorphe sous l'addition, réutilisation sécure de l'aléat et réutilisation sécure des clées. La propriété homomorphe de l'encryption fait en sorte que $\mathbf{xC}$ est l'encryption de $\mathbf{xM}$. Pour la fonction injective, on pourrait utiliser $\mathbf{M} = \mathbf{I}$ de façon à ce que $\mathbf{xC}$ se décrypte en $\mathbf{xI} = \mathbf{x}$. Avec $\mathbf{M} = \mathbf{0}$ on aurait que la décryption de $\mathbf{xC}$ est $\mathbf{0}$ donc la plupart de l'information sur $\mathbf{x}$ est perdue. Insistons cependant sur le fait que pour la fonction lossy, $\mathbf{xC}$ peut tout de même laisser s'échapper beaucoup informations sur $\mathbf{x}$. En particulier, lors de la multiplication $\mathbf{xC}$, les termes d'erreurs des cyptogrammes de $\mathbf{C}$ sont amplifiés par $\mathbf{x}$ et l'erreur $\mathbf{xE}$ résultante peut donner des informations sur $\mathbf{x}$. Plus formellement, le terme d'erreur des cryptogrammes vient ajouter un degré de liberté suplémentaire à la sortie de la fonction et si on n'addresse pas ce problème l'image de la fonction lossy sera trop grande (et donc pas assez lossy). Nous montrerons dans la section suivante comment résoudre ce problème en réduisant le nombre de cryptogrammes générés.

9.3 Construction et sécurité

Premièrement, définissons un schéma d'encryption à clé privée basé sur LWE semblable à celui de Regev et qui sera au centre de la construction. Notons que $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ est le groupe de nombres réels $[0, 1)$ avec l'addition modulo 1.

— Le message est un entier $m \in \mathbb{Z}_p$
— La clée secrète est $sk := \mathbf{z} \in \mathbb{Z}_q^d$
— Pour encrypter $m \in \mathbb{Z}_p$, choisir uniformément $\mathbf{a} \in \mathbb{Z}_q^d$ et $e \in \chi$ (la distribution d'erreur sera définie plus tard). Définissons $c_m = \frac{m}{p} \in \mathbb{T}$. Le cryptogramme est alors

$$E_{\mathbf{z}}(m, \mathbf{a}, e) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{z} \rangle + \lfloor q c_m \rceil + e) \in \mathbb{Z}_q^d \times \mathbb{Z}_q$$

— Pour décrypter un cryptogramme $c = (\mathbf{a}, c')$, $D_{\mathbf{z}}(c)$ calcule

$$t = \frac{c' - \langle \mathbf{a}, \mathbf{z} \rangle}{q} = \frac{\lfloor q c_m \rceil + e}{q} \in \mathbb{T}$$

et retourne $m' \in \mathbb{Z}_p$ tel que $t - c_{m'} = \frac{\lfloor \frac{qm}{p} \rceil - \frac{qm'}{p} + e}{q} \in \mathbb{T}$ soit le plus proche de 0 modulo 1. Notons que pour que la décryption soit correcte on doit avoir $|\lfloor \frac{qm}{p} \rceil - \frac{qm'}{p} + e| < \frac{q}{2p}$.
— Ce système est homomorphe sous l'addition car

$$E_{\mathbf{z}}(m, \mathbf{a}, e) + E_{\mathbf{z}}(m', \mathbf{a}', e') = E_{\mathbf{z}}(m + m', \mathbf{a} + \mathbf{a}', e + e')$$

Ensuite, voici une variante pour en encrypter des matrices.

— Le message est une matrice $\mathbf{M} \in \mathbb{Z}_p^{h \times w}$
— La clée secrète est $sk := \mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_w)$ où $\mathbf{z}_i \in \mathbb{Z}_q^d$ est la clée utilisée pour encrypter la i-ème colonne.
— Pour encrypter $\mathbf{M} \in \mathbb{Z}_p^{h \times w}$, procéder de la façon suivante. Pour chaque rangée choisir $\mathbf{a}_i \leftarrow \mathbb{Z}_q^d$ uniformément, ainsi formant une matrice d'aléat $\mathbf{A} \in \mathbb{Z}_q^{h \times d}$. Ensuite, générer une matrice d'erreur $\mathbf{E} \in \mathbb{Z}_q^{h \times w}$ où chaque $e_{i,j} \leftarrow \chi$. Finalement, chaque élément de la matrice encryptée $\mathbf{C} = E_{\mathbf{Z}}(\mathbf{M}, \mathbf{A}, \mathbf{E})$ est défini comme

$$c_{i,j} = E_{\mathbf{z}_j}(m_{i,j}, \mathbf{a}_i, e_{i,j}) = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{z}_j \rangle + \lfloor q c_{m_{i,j}} \rceil + e_{i,j})$$

Les éléments de la matrice sont donc encryptés avec la clée associée à leur colonne, l'aléat associé à leur rangée et une erreur unique.
— Pour décrypter, on calcule $\mathbf{M} = D_{\mathbf{Z}}(\mathbf{C})$ où $m_{i,j} = D_{\mathbf{z}_j}(c_{i,j})$.
— Ce système est homomorphe sous l'addition cryptogamme et la multiplication par un vecteur : Si $\mathbf{C} = E_{\mathbf{Z}}(\mathbf{M}, \mathbf{A}, \mathbf{E})$, alors $\mathbf{xC} = E_{\mathbf{Z}}(\mathbf{xM}, \mathbf{xA}, \mathbf{xE})$ pour tout $\mathbf{x} \in \mathbb{Z}_p^h$.

Comme mentionné plus haut, les terme d'erreur contenus dans les cryptogrammes de $\mathbf{xC}$ peuvent laisser fuire de l'information à propos de $\mathbf{x}$ dans le cas lossy (matrice $\mathbf{0}$) à cause de leur amplification (on pourrait aussi dire que cela rends l'image de la fonction lossy trop grande). Pour règler ce problème, on réduira le nombre de cryptogrammes générés par la fonction en choisisant des matrices

qui compressent $\mathbf{x}$ tout en gardant les bonnes propriétés. Nous définirons ces matrices et discuterons de leur fonctionnement par la suite.

— Soit $l = \lfloor \log p \rfloor$ et $m = n/l$ (sans perte de généralité n est divisible par l). La matrice de la fonction injective est

$$\mathbf{B} = \begin{bmatrix} \mathbf{b} & 0 & \dots & 0 \\ 0 & \mathbf{b} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{b} \end{bmatrix} \in \mathbb{Z}_p^{n \times m}$$

où $\mathbf{b} = (2^0, 2^1, \dots, 2^{l-2}) \in \mathbb{Z}_p^{l \times 1}$

— La matrice de la fonction lossy est $\mathbf{0} \in \mathbb{Z}_p^{n \times m}$

On peut voir qu'avec des matrices, le produit $\mathbf{xC}$ se trouve dans $\mathbb{Z}_p^m$ donc la sortie de la fonction comporte moins de cryptogrammes ($m < n$). Pour la fonction lossy, cela veut dire que moins d'informations peuvent être transmise par les termes d'erreur (autrement dit, la grandeur de l'image de la fonction peut être controllée par le paramètre p). De plus la structure de $\mathbf{B}$ fait en sorte que $\mathbf{xB}$ est toujours injectif. En effet, $\mathbf{v} = \mathbf{xB}$ représente la valeur modulo p des l blocs de $\mathbf{x}$ donc chaque $\mathbf{x}$ est associé à un unique $\mathbf{v}$. Ainsi, l'image de la LTDF est $\{0,1\}^n$ et son domaine est $\mathbb{Z}_p^m$. Formellement, on a le théorème suivant :

Théorème 10. *Soit $c_1 > 0, c_2 > 1$ et $c_3 > 1$ définissant les paramètres de la façon suivante : $p = n^{c_1}$, $q \in [4pn, O(pn^{c_2})]$, $n = d^{c_3}$ et χ la distribution normale sur $\mathbb{T}$ avec moyenne 0 et écart-type $\alpha/\sqrt{2\pi}$ où $\alpha \leq 1/(32pn)$. Alors, la construction définie plus haut est presque-toujours*[1] *une collection de (n,k)-LTDF sous l'hypothèse que $LWE_{q,\chi}$ est difficle. On également $r \leq (\frac{c_2}{c_1} + o(1)) \cdot n$*

On peut ainsi manipuler c_1 et c_2 pour obtenir une fonction lossy qui perd un pourcentage arbitraire de l'entré (ex. 99%). Notons également qu'il existe une construction analogue basée sur ring-LWE et plus efficace.

Or, pour que la difficultée de $LWE_{q,\chi}$ repose sur celle de GapSVP en pire cas, on doit avoir $q > 2\frac{\sqrt{d}}{\alpha}$ (voir 2.1.2). En fixant $\alpha = 1/(32pn)$ on a donc que

$$q > 64pn\sqrt{d} = 64pn^{1+1/(2c_3)}.$$

On voit donc qu'il faut poser $c_2 = 1 + 1/(2c_3)$ et choisir $q = O(pn^{c_2})$.
On a donc que la construction est sécure basée sur GapSVP avec un facteur d'approximation $\tilde{O}(d/\alpha) = \tilde{O}(d^{1+c_3(c_1+1)})$, problème conjecturé difficile même pour les ordinateurs quantique.

1. Signifie que la $S_{inj}(1^\lambda)$ décrit une fonction injective sauf avec probabilité négligeable. Cela est purement technique et n'a aucun impact sur les applications.

Références

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In : *Journal of the ACM* (2005).

[LM06] Vadim Lyubashevsky et Daniele Micciancio. "Generalized compact knapsacks are collision resistant". In : *ICALP* (2006).

[MR07] Daniele Micciancio et Oded Regev. "Worst-case to average-case reductions based on Gaussian measures". In : *SIAM Journal on Computing* (2007).

[LM09] Vadim Lyubashevsky et Daniele Micciancio. "On bounded distance decoding, unique shortest vectors, and the minimum distance problem". In : *CRYPTO* (2009).

[LPR10] Vadim Lyubashevsky, Chris Peikert et Oded Regev. "On ideal lattices and learning and learning with errors over rings". In : *EUROCRYPT* (2010).

[Bra+13] Zvika Brakerski et al. "Classical Hardness of Learning with Errors". In : (2013).