# Projet cryptologie

Dylan Mona, M1 MIC
Université Paris-Cité
Année 2024–2025

## Contents

# 1 Le problème Learning With Errors (LWE)

Soient $n \geq 1$ et $q \geq 2$ deux entiers, ainsi que $\alpha \in ]0, 1[$.

Pour un vecteur secret $s \in \mathbb{Z}_q^n$, un échantillon LWE est généré comme suit :

- On choisit un vecteur $a \in \mathbb{Z}_q^n$ uniformément au hasard.
- On tire une erreur $e$ suivant une distribution gaussienne discrète $D_{\mathbb{Z},\alpha q}$.
- On calcule $b = \langle a, s \rangle + e \mod q$.

On obtient ainsi un échantillon $(a, b) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Il existe deux versions du problème LWE, qui sont équivalentes :

- **Version calculatoire :** étant donné plusieurs échantillons LWE $(a_i, b_i)$, retrouver le secret $s \in \mathbb{Z}_q^n$.
- **Version décisionnelle :** distinguer des échantillons LWE $(a, b = \langle a, s \rangle + e)$ d'échantillons uniformes $(a, b) \in U(\mathbb{Z}_q^{n+1})$.

vous pouvez retrouver un code qui permet de comparer une géneration lwe et uniforme (lwevsU.py)

# 2 Équivalence entre les versions décisionnelle et calculatoire du LWE

## 2.1 De la version calculatoire à la version décisionnelle

Supposons que nous disposons d'un algorithme capable de résoudre la version calculatoire, c'est-à-dire retrouver $s$ à partir d'échantillons LWE.

Étant donné un lot de couples $(a_i, b_i)$ :

- On utilise l'algorithme pour calculer un candidat $s' \in \mathbb{Z}_q^n$.
- Pour chaque échantillon, on calcule $e'_i = b_i - \langle a_i, s' \rangle \mod q$.

Si les valeurs $e'_i$ sont toutes proches de 0 modulo $q$, alors les échantillons suivent la distribution LWE. Sinon, ils sont statistiquement proches de l'uniforme.

Donc on peut distinguer des échantillons LWE de données aléatoires. Ainsi, la version calculatoire implique la version décisionnelle.

vous pouvez retouver un code pour visualiser ceci (c d.py)

## 2.2 De la version décisionnelle à la version calculatoire

Supposons maintenant que nous disposons d'un oracle $D$ capable de résoudre le problème décisionnel.

Nous allons reconstruire $s = (s_1, \ldots, s_n)$ coordonnée par coordonnée.

Pour chaque $i \in \{1, \ldots, n\}$ :

- on essaye tout les possibilité $s_i^* \in \mathbb{Z}_q$ et on tire un $u \leftarrow U(\mathbb{Z}_q)$.

- On modifie un échantillon $(a, b)$ pour obtenir :

$$a' = a + u \cdot e_i, \quad b' = b + u \cdot s_i^*$$

où $e_i$ est le vecteur avec un 1 en position $i$ et 0 ailleurs.

**Cas 1 :** Si $s_i^* = s_i$, alors :

$$\begin{aligned} a' &= a + ue_i \\ \langle a', s\rangle &= \langle a, s\rangle + us_i \\ b' &= b + us_i = \langle a, s\rangle + e + us_i = \langle a', s\rangle + e \end{aligned}$$

Donc $(a', b')$ est un échantillon LWE.

**Cas 2 :** Si $s_i^* \neq s_i$, alors :

$$\begin{aligned} a' &= a + ue_i \\ b' &= b + us_i^* = \langle a, s\rangle + e + us_i^* \\ &= \langle a', s\rangle + e + u(s_i^* - s_i) \end{aligned}$$

Comme $u(s_i^* - s_i)$ est uniforme sur $\mathbb{Z}_q$, alors $(a', b')$ est proche de l'uniforme.

on utilise l'oracle $D$ pour distinguer quelle hypothèse $s_i^*$ donne un échantillon LWE. Celle-ci est probablement égale à la vraie valeur $s_i$.

On répète pour chaque $i$, ce qui donne un coût de $O(nq)$ appels à $D$.

Donc la version décisionnelle permet de retrouver le secret, donc elle implique la version calculatoire.

vous pouvez retouver un code pour visualiser ceci (d c.py)

# 3 Chiffrement de Regev

Soient $n, m, q$ des entiers avec $q$ premier et $m \geq 4(n+1)\log_2 q$, et soit $\alpha$ un réel dans l'intervalle $]0, 1/(4m)[$.

## 3.1 KeyGen (Génération de clés)

- La clé secrète est $s \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$.
- La clé publique est un couple $(A, b) = (A, As + e)$, où :
  - $A \leftarrow \mathcal{U}(\mathbb{Z}_q^{m\times n})$,
  - $e \leftarrow D_{\mathbb{Z}^m, \alpha q}$ (distribution gaussienne discrète).

## 3.2 Enc (Chiffrement)

- Étant donné un message $M \in \{0, 1\}$ :
  - Échantillonner $r \leftarrow \mathcal{U}(\{0, 1\}^m)$,
  - Retourner le chiffré $(u^T, v) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ défini par :

$$u^T = r^T A, \quad v = r^T b + \left\lfloor \frac{q}{2} \right\rfloor \cdot M$$

### 3.3 Dec (Déchiffrement)

- Étant donné un chiffré $(u^T, v) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ :
  - Calculer $v - u^T s$,
  - Le message clair est :

$$\begin{cases} 0 & \text{si le résultat est plus proche de 0 que de } \left\lfloor \frac{q}{2} \right\rfloor, \\ 1 & \text{sinon.} \end{cases}$$

- En effet, on a :

$$\begin{aligned} v - u^T s &= r^T b + \left\lfloor \frac{q}{2} \right\rfloor \cdot M - r^T A s \\ &= r^T (As + e) + \left\lfloor \frac{q}{2} \right\rfloor \cdot M - r^T A s \\ &= r^T A s + r^T e + \left\lfloor \frac{q}{2} \right\rfloor \cdot M - r^T A s \\ &= \left\lfloor \frac{q}{2} \right\rfloor \cdot M + r^T e \end{aligned}$$

- Le terme d'erreur $r^T e$ est petit, donc la quantité est proche de 0 si $M = 0$, et proche de $\left\lfloor \frac{q}{2} \right\rfloor$ si $M = 1$.

vous pouvez retrouver un code pour visualiser ceci (regev.py)

## 4 outils LWE

Une base courte du noyau modulo $q$ de $A$, c'est-à-dire :

$$\Lambda_q^\perp(A) = \{x \in \mathbb{Z}^m \mid A^T x = 0 \mod q\}$$

### 4.1 Leftover Hash Lemma

Supposons que $m \geq 4n \log_2 q$ et que $q$ est un nombre premier. Soient $A \leftarrow \mathcal{U}(\mathbb{Z}_q^{n \times m})$ et $r \leftarrow \mathcal{U}(\{0,1\}^m)$. Alors la distribution de la paire $(A, Ar)$ est à distance statistique au plus $2^{-n}$ de la distribution uniforme $\mathcal{U}(\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n)$.

### 4.2 genbasis

Un algorithme probabiliste polynomial qui prend en entrée des entiers $n$, $m$ et $q \geq 2$ (écrits en unaire), tels que $m$ est significativement plus grand que $n$ et $q$, et qui renvoie deux matrices $A \in \mathbb{Z}_q^{m \times n}$ et $S \in \mathbb{Z}^{m \times m}$ telles que :

- la distribution de $A$ est à distance statistique négligeable de la distribution uniforme, - les lignes $s_i$ de $S$ forment une base de $\Lambda_q^\perp(A)$, - et $\max_i \|s_i\| \leq O(\sqrt{n} \log q)$.

### 4.3 Retrouver le secret $s$ à partir de LWE grâce à la trappe

Soit un échantillon LWE donné par :

$$b = As + e \mod q$$

où :

- $A \in \mathbb{Z}_q^{m \times n}$ est une matrice aléatoire (avec une trappe $S$),
- $s \in \mathbb{Z}_q^n$ est le secret,
- $e \in \mathbb{Z}^m$ est un vecteur de bruit "petit",
- $b \in \mathbb{Z}_q^m$.

Si on connaît la trappe $S$, on peut multiplier à gauche l'équation par $S$ :

$$Sb = S(As + e) = SAs + Se \mod q.$$

Mais, par définition de la trappe, $SA = 0 \mod q$ car $S$ est une base du noyau modulo $q$, donc :

$$Sb = Se \mod q.$$

Comme $e$ est petit et que les vecteurs de $S$ sont courts, les composantes de $Se$ sont bien inférieures à $q$. Ainsi, on a en réalité :

$$Sb = Se.$$

Ensuite, on peut retrouver $e$ en résolvant :

$$e = S^{-1}Sb,$$

où $S^{-1}$ désigne l'inverse (ou une méthode de décomposition permettant de retrouver $e$).

Enfin, on obtient :

$$As = b - e,$$

ce qui permet de résoudre un système sans bruit pour retrouver le secret $s$.

## 4.4 Échantillonnage Gaussien

On souhaite échantillonner un vecteur $x \in \mathbb{Z}^m$ selon une distribution gaussienne discrète de paramètre s assez grand par rapport au paramètre (n q m) tel que :

$$x^T A = y^T \mod q.$$

tout d'abord on résoudre l'équation suivante :

$$x_0^T A = y^T \mod q$$

en trouvant une solution quelconque $x_0$.

Ensuite, on échantillonne autour de $-x_0$ dans le réseau dual :

$$x_1 \leftarrow D_{\Lambda_q^\perp(A), s, -x_0}$$

où $D_{\Lambda_q^\perp(A), s, -x_0}$ désigne une distribution gaussienne discrète centrée en $-x_0$ sur le réseau $\Lambda_q^\perp(A)$, avec paramètre $s$.

On retourne :

$$x = x_0 + x_1$$

Alors on a :

$$x^T A = (x_0 + x_1)^T A = x_0^T A + x_1^T A = y^T + 0 \mod q = y^T \mod q,$$

et $x$ suit une distribution gaussienne centrée sur la solution $x_0$.

### 4.5 ExtBasis

Soit une matrice $B \in \mathbb{Z}_q^{m' \times n}$, obtenue en ajoutant des lignes à $A$. Si l'on possède une trappe $S$ pour $A$, on peut étendre cette trappe à la matrice $B$.

Pour chaque nouvelle ligne $b_i$ ajoutée à $A$, on effectue les étapes suivantes :

1. Trouver un vecteur $s_i$ tel que :
$$-s_i^T A = b_i \mod q$$
2. Construire $s_i$ à l'aide de l'algorithme d'échantillonnage Gaussien (comme décrit ci-dessus).
3. Construire une grande matrice trappe :
$$T = \begin{bmatrix} S & 0 \\ s_{m+1}^T & \\ \vdots & \mathbb{I}_{m'-m+1} \\ s_{m'}^T & \end{bmatrix}$$

ou la matrcie $\mathbb{I}_{m'-m+1}$ est la matrice identité de taille m'-m+1.

La matrice $T$ ainsi construite satisfait :

$$TB = 0 \mod q$$

et constitue une base complète du nouveau réseau $\Lambda_q^{\perp}(B)$.

# 5 Chiffrement par attribut

## 5.1 Two to One Recoding (TOR)

Le recodage de type "Two to One" (TOR) est un outil utilisé dans les schémas de chiffrement fondés sur les réseaux, notamment dans le chiffrement basé sur les attributs.

Supposons que l'on dispose de trois matrices publiques :

$$A_0, A_1, A_{\text{tgt}} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n}),$$

L'Objectif du TOR est de trouver deux matrices $R_0, R_1 \in \mathbb{Z}^{m \times m}$ à coefficients petits, telles que :

$$R_0 A_0 + R_1 A_1 = A_{\text{tgt}} \mod q.$$

Ces matrices sont appelées matrices de rechiffrement de $A_0$ et $A_1$ vers $A_{\text{tgt}}$.

Si l'on dispose de deux vecteurs de la forme :

$$A_0 s + e_0 \quad \text{et} \quad A_1 s + e_1 \in \mathbb{Z}_q^m,$$

qui encodent le même secret $s$, alors on peut former :

$$R_0(A_0 s + e_0) + R_1(A_1 s + e_1) = A_{\text{tgt}} s + e_{\text{tgt}},$$

avec :

$$e_{\text{tgt}} = R_0 e_0 + R_1 e_1.$$

Ce résultat représente un nouvel encodage de $s$ sous la matrice $A_{\text{tgt}}$, avec un bruit $e_{\text{tgt}}$ toujours petit comme $R_0, R_1, e_0, e_1$ le sont.

### 5.1.1 Méthodes pour calculer $R_0$ et $R_1$

1. **Avec une trappe pour $A_0$ :**
   - On suppose connaître une base courte $S_0$ du noyau $\Lambda_q^\perp(A_0)$.
   - On échantillonne aléatoirement $R_1 \leftarrow D_{\mathbb{Z}^{m\times m},s}$.
   - Ensuite, pour chaque ligne de $R_0$, on utilise l'échantillonnage Gaussien pour résoudre :
   $$R_0 A_0 = A_{\text{tgt}} - R_1 A_1 \mod q.$$
2. **Avec une trappe pour $A_1$ :**
   - Cas symétrique : on choisit $R_0$ aléatoirement, et on utilise la trappe de $A_1$ pour construire $R_1$ via l'échantillonnage Gaussien.
3. **Sans trappe (simulation) :**
   - On échantillonne :
   $$R_0 \leftarrow D_{\mathbb{Z}^{m\times m},s}, \quad R_1 \leftarrow D_{\mathbb{Z}^{m\times m},s},$$
   - Puis on définit directement :
   $$A_{\text{tgt}} := R_0 A_0 + R_1 A_1 \mod q.$$
   - Ce cas est utilisé dans les simulations (ex. : preuves de sécurité) où aucune trappe n'est requise.

# 6 Protocole de Gorbunov, Vaikuntanathan et Wee (GVW)

## 6.1 Circuits booléens

Un circuit booléen $C$ est constitué de portes logiques $g : \{0,1\}^2 \to \{0,1\}$ et d'arêtes portant des valeurs dans $\{0,1\}$.

- On suppose que le circuit comporte exactement $\ell$ arêtes d'entrée : $w_1, \ldots, w_\ell$.
- Le circuit possède une arête de sortie notée $w_{|C|}$, où $|C|$ désigne le nombre total d'arêtes du circuit.

Chaque arête $w_i$ qui n'est pas une arête d'entrée (c'est-à-dire pour $\ell < i \leq |C|$) est une sortie d'une porte logique $g_i$, laquelle prend en entrée deux arêtes $w_{\text{inp}_0(i)}$ et $w_{\text{inp}_1(i)}$, avec :
$$\text{inp}_0(i) \leq \text{inp}_1(i) < i.$$
vous pouvez retouver un code pour visualiser ceci (attribut.py)

**Évaluation du circuit :** Si les valeurs portées par $w_{\text{inp}_0(i)}$ et $w_{\text{inp}_1(i)}$ sont respectivement $b_0$ et $b_1$, alors la valeur portée par $w_i$ est :
$$w_i = g_i(b_0, b_1).$$

Ainsi, l'évaluation du circuit s'effectue de manière séquentielle en suivant l'ordre topologique des arêtes, de l'entrée vers la sortie.

## 6.2 KeyGen (Génération de clés)

Chaque attribut $\mathrm{att}_i \in \{0,1\}$ est binaire. Pour chaque bit possible $b \in \{0,1\}$, on génère :

- une matrice publique $A_i^{(b)} \in \mathbb{Z}_q^{m \times n}$,
- une trappe (base courte) $S_i^{(b)} \in \mathbb{Z}_q^{m \times m}$,

à l'aide de l'algorithme genbasis.
De plus, on génère une matrice de sortie :

$$A_{\mathrm{out}} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n})$$

Cette matrice correspond à l'arête de sortie du circuit logique.

**Clé publique (mpk) :**

$$\mathrm{mpk} = \{A_i^{(b)} \mid i \leq \ell, b \in \{0,1\}\} \cup \{A_{\mathrm{out}}\}$$

**Clé secrète (msk) :**

$$\mathrm{msk} = \{S_i^{(b)} \mid i \leq \ell, b \in \{0,1\}\}$$

## 6.3 Enc (Chiffrement)

L'expéditeur souhaite chiffrer un bit $M \in \{0,1\}^m$ sous les attributs $(\mathrm{att}_1, \ldots, \mathrm{att}_\ell)$.

- Échantillonner :
  - $s \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ — vecteur aléatoire,
  - $e_1, \ldots, e_\ell, e_{\mathrm{out}} \leftarrow D_{\mathbb{Z}^m, \alpha q}$ — bruit gaussien discret.
- Pour chaque $i$, calculer :
$$c_i = A_i^{(\mathrm{att}_i)} \cdot s + e_i$$
- Calculer le chiffré de sortie :
$$c_{\mathrm{out}} = A_{\mathrm{out}} \cdot s + e_{\mathrm{out}} + \left\lfloor \frac{q}{2} \right\rfloor \cdot M$$

Le chiffré global est donc :

$$\mathrm{ct} = (c_1, \ldots, c_\ell, c_{\mathrm{out}})$$

## 6.4 Génération de la clé de déchiffrement

- Pour chaque arête $i > l$, on génere comme fait pour la KeyGen :
$$(A_i^{(0)}, S_i^{(0)}), \quad (A_i^{(1)}, S_i^{(1)})$$
- Pour l'arête de sortie $|C|$ :
$$A_{|C|}^{(1)} = A_{\mathrm{out}}, \quad A_{|C|}^{(0)} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m \times n})$$

- Pour chaque porte logique $g_i$, dont les entrées sont les arêtes $w_{\text{inp}_0(i)}$, $w_{\text{inp}_1(i)}$, on construit :
  - Des matrices de rechiffrement :
$$R_i^{(b_0,b_1)} \in \mathbb{Z}_q^{m\times 2m}$$
  - Satisfaisant :
$$R_i^{(b_0,b_1)} \cdot \begin{bmatrix} A_{\text{inp}_0(i)}^{(b_0)} \\ A_{\text{inp}_1(i)}^{(b_1)} \end{bmatrix} = A_i(g_i(b_0,b_1)) \mod q$$

La clé de déchiffrement est :
$$\text{sk}_C = \{R_i^{(b_0,b_1)}\}_{i,b_0,b_1}$$

## 6.5 Dec (Déchiffrement)

Le récepteur évalue le circuit booléen $C$ sur les attributs et les chiffrés associés.

- Pour chaque porte logique $g_i$, avec entrées $w_{\text{inp}_0(i)}, w_{\text{inp}_1(i)}$, on note :
  - $b_0, b_1 \in \{0,1\}$ les résultats logiques partiels,
  - $c_{\text{inp}_0(i)} = A_{\text{inp}_0(i)}^{(b_0)} \cdot s + e_{\text{inp}_0(i)}$,
  - $c_{\text{inp}_1(i)} = A_{\text{inp}_1(i)}^{(b_1)} \cdot s + e_{\text{inp}_1(i)}$.
- On applique la matrice de rechiffrement :
$$c_i = R_i(b_0,b_1) \cdot \begin{bmatrix} c_{\text{inp}_0(i)} \\ c_{\text{inp}_1(i)} \end{bmatrix} = A_i(g_i(b_0,b_1)) \cdot s + e_i$$
- À la sortie du circuit, notée $|C|$, on obtient :
$$c_{|C|} = A_{|C|}^{(j)} \cdot s + e_{|C|} \quad \text{avec } j \in \{0,1\}$$

**Si** $C(\mathbf{att}_1, \ldots, \mathbf{att}_\ell) = 1$ **:** On a :
$$c_{|C|} = A_{\text{out}} \cdot s + e_{|C|}$$
$$c_{\text{out}} = A_{\text{out}} \cdot s + e_{\text{out}} + \left\lfloor \frac{q}{2} \right\rfloor \cdot M$$
$$c_{\text{out}} - c_{|C|} = (e_{\text{out}} - e_{|C|}) + \left\lfloor \frac{q}{2} \right\rfloor \cdot M = v$$

Comme $e_{\text{out}} - e_{|C|}$ est petit, on peut déduire $m$ en examinant chaque composante $v_j$ :
$$v_j \approx 0 \mod q \Rightarrow M_j = 0, \quad v_j \approx \left\lfloor \frac{q}{2} \right\rfloor \mod q \Rightarrow M_j = 1$$

**Si** $C(\mathbf{att}_1, \ldots, \mathbf{att}_\ell) = 0$ **:** On a :
$$c_{|C|} = A' \cdot s + e_{|C|} \quad \text{avec } A' \neq A_{\text{out}}$$
$$c_{\text{out}} - c_{|C|} = (A_{\text{out}} - A') \cdot s + (e_{\text{out}} - e_{|C|}) + \left\lfloor \frac{q}{2} \right\rfloor \cdot M$$

Puisque $(A_{\text{out}} - A') \cdot s$ est non négligeable, il devient impossible de récupérer $M$ de façon fiable.

vous pouvez retouver un code pour visualiser ceci (att.py)