

Cryptographie post quantique : TP1 - Chiffrement basé sur les réseaux euclidiens

0.1 Paquetages nécessaires

On vous joint ici un ensemble de paquetages sage pouvant vous être utile lors de vos implémentations.

- [Distributions Gaussiennes Discrètes](#)
- [LWE](#)

On rappelle également les différentes notations décrites dans ce TP:

- On dénote par $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ l'ensemble des entiers modulo q ,
- $x \xleftarrow{\$} S$ décrit un tirage uniforme dans l'ensemble fini S .
- $x \leftarrow \mathcal{D}_{S,c,\sigma}$ décrit une distribution gaussienne dans l'ensemble S d'écart-type σ , centrée en c . Par défaut la distribution est centrée en 0.

1 Distributions de probabilités

(UniformeFR) **Exercice 1.**

Loi uniforme

Etudier la distribution uniforme: tirer m éléments de $\mathbb{Z}_q$ et visualiser la distribution en sortie avec [plot](#).

1. Pour $q = 3$, et $m = 10^2, 10^3, 10^4$.
2. Pour $q = 10$, et $m = 10^2, 10^3, 10^4$.
3. Pour $q = 100$, et $m = 10^3, 10^4, 10^5$.

(GaussienneFR) **Exercice 2.**

Loi gaussienne

Etudier la distribution gaussienne: tirer m éléments de $\mathcal{D}_{\mathbb{R},\sigma}$ et visualiser la distribution en sortie avec [plot](#).

1. Pour $\sigma = \sqrt{2}, m = 10^3$
2. Pour $\sigma = 10, m = 10^3$
3. Pour $\sigma = 10^3, m = 10^3$

(GaussienneDiscreteFR) **Exercice 3.**

Loi gaussienne discrète

Etudier la distribution gaussienne discrète: tirer m éléments de $\mathcal{D}_{\mathbb{Z},\sigma}$ et visualiser la distribution en sortie avec [plot](#).

1. Pour $\sigma = \sqrt{2}, m = 10^3$
2. Pour $\sigma = 10, m = 10^3$
3. Pour $\sigma = 10^3, m = 10^3$

2 Schémas de chiffrement fondamentaux basés sur les réseaux

(RegevFR) **Exercice 4.**

Chiffrement de Regev

Oded Regev, « On lattices, learning with errors, random linear codes, and cryptography », In Proc. of STOCs, 2005

Le premier schéma à implémenter est le schéma de chiffrement proposé par Regev en tant que premier cas d'usage de l'hypothèse LWE.

1. Implémenter une fonction **RegevGenerator** prenant en entrée un paramètre de sécurité λ construisant une variable **params** comprenant les différentes variables publiques fixées avant la génération des clés.

Pour l'instant, faites en sorte à ce que les variables soient fixées à des valeurs arbitraires, mais valides satisfaisant:

- la taille du secret LWE est $n = \lambda$,
- le modulo q est un nombre entier premier,
- le nombre d'échantillons LWE est $m > n \log q$,
- l'écart type de la distribution gaussienne discrète de l'erreur est $\sigma = 3$.

Afin d'assurer l'optimalité, on choisira les plus petits paramètres satisfaisant chacune des conditions.

2. Implémenter la fonction **RegevKeyGen** d'entrée **params** générant à la fois une clé privée ainsi qu'une clé publique associée. On rappelle la définition de la fonction de génération de clés, elle sort:

$$\text{sk} = \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n; \text{pk} = (\text{pk}_1, \text{pk}_2) = (\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \text{ avec } \mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \text{ et } \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z}, \sigma}^m.$$

3. Implémenter la fonction **RegevEncrypt** d'entrée **params**, **pk** et $b \in \{0, 1\}$ qui chiffre un bit b en utilisant la clé publique **pk** et ressortant un chiffré **c**. La fonction se déroule en deux étapes:

1. Échantillonner un vecteur **r** comprenant m bits aléatoires i.e. $\mathbf{r} \xleftarrow{\$} \{0, 1\}^m$.
2. Construire le chiffré $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1) := (\mathbf{r}^\top \text{pk}_1, \mathbf{r}^\top \text{pk}_2 + \lfloor \frac{q}{2} \rfloor \cdot b)$.

4. Implémenter la fonction **RegevDecrypt** qui, en recevant un chiffré $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1)$ ressort 0 si $\mathbf{c}_1 - \mathbf{c}_0^\top \text{sk}$ est plus proche de 0 que de $\lfloor \frac{q}{2} \rfloor$ et 1 sinon.
5. Tester expérimentalement que votre implémentation est correcte i.e. que le déchiffrement d'un chiffré donne bien le bon clair choisi initial pour $\lambda = 16, 32, 64$.
6. Étendre la définition du chiffrement pour qu'elle puisse prendre en entrée un vecteur de bits $b \in \{0, 1\}^\ell$ pour $\ell > 1$. Modifier les fonctions **RegevEncrypt** et **RegevDecrypt** en conséquence.

Astuce: essayer de chiffrer un vecteur de bits en utilisant la même clé publique **pk** et un vecteur différent pour chaque bit du vecteur. Écrire sous forme matricielle le chiffré $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1)$ d'un message clair $\mathbf{b} \in \{0, 1\}^\ell$.

(DualRegevFR) **Exercice 5.**

Chiffrement Dual-Regev

C. Gentry, C. Peikert, and V. Vaikuntanathan. "Trapdoors for hard lattices and new cryptographic constructions." In Proc. of STOC, pages 197–206. ACM, 2008

Le chiffrement Dual-Regev est une autre proposition d'un schéma de chiffrement basé sur LWE par Gentry Peikert et Vaikuntanathan. L'idée derrière est sensiblement la même mais avec le vecteur de m bits en tant que clé secrète et le secret LWE $\mathbf{s}$ est renouvelé à chaque chiffrement. La sécurité de ce chiffrement ne dépend cependant pas uniquement de LWE mais également d'une instance de Leftover Hash Lemma. Afin d'assurer l'optimalité des paramètres, on choisira le plus petit paramètre satisfaisant chacune des conditions.

1. Implémenter une fonction **DualGenerator** prenant en entrée un paramètre de sécurité λ construisant une variable **params** comprenant les différentes variables publiques fixées avant la génération des clés.

Pour l'instant, faites en sorte à ce que les variables soient fixées à des valeurs arbitraires, mais valides satisfaisant:

- la taille du secret LWE est $n = \lambda$,
- le modulo q est un nombre entier premier,
- le nombre d'échantillons LWE est $m > n \log q$,
- l'écart type de la distribution gaussienne discrète de l'erreur est $\sigma = 3$.

Afin d'assurer l'optimalité, on choisira les plus petits paramètres satisfaisant chacune des conditions.

2. Implémenter la fonction **DualKeyGen** d'entrée **params** générant à la fois une clé privée ainsi qu'une clé publique associée. La clé privée $\mathbf{sk}$ est un vecteur contenant m bits uniformément tiré. La clé publique associée $\mathbf{pk}$ est définie par $\mathbf{pk} = \mathbf{sk}^\top \mathbf{A}$.
3. Implémenter la fonction **DualEncrypt** d'entrée **params**, **pk** et $b \in \{0,1\}$ qui chiffre un bit b en utilisant la clé publique $\mathbf{pk}$ et ressortant un chiffré $\mathbf{c}$. On rappelle la fonction:
 1. Construire un tuple $\text{LWE}_{n,m,q,\sigma}$, dénoté $\mathbf{b}$ avec les éléments secrets $\mathbf{s}$ et $\mathbf{e}$. En reprenant la définition de LWE, dans quel ensemble doivent-il être échantillonnés ?
 2. Construire le chiffré $\mathbf{c} = (c_1, c_2) := \left(\mathbf{b}, \mathbf{pk}^\top \mathbf{s} + e' + \left\lfloor \frac{q}{2} \right\rfloor \cdot b \right)$ avec une erreur $e' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$.
4. Implémenter la fonction **DualDecrypt** qui, en recevant, un chiffré $\mathbf{c} = (c_1, c_2)$ ressort 0 si $c_2 - c_1^\top \mathbf{sk}$ est plus proche de 0 que de $\left\lfloor \frac{q}{2} \right\rfloor$ et 1 sinon.
5. Tester expérimentalement que votre implémentation est correcte i.e. que le déchiffrement d'un chiffré donne bien le bon clair choisi initial pour $\lambda = 16, 32, 64$.
6. Etendre la définition du chiffrement pour qu'elle puisse prendre en entrée un vecteur de bits $b \in \{0,1\}^\ell$ pour $\ell > 1$. Modifier les fonctions en conséquence.

3 Aller plus loin

(EfficaciteFR) **Exercice 6.**

Analyse de l'efficacité des deux schémas

Vous pouvez faire les tests pour la différence de vitesse en utilisant `%timeit` dans un notebook.

1. Construire un jeu permettant d'évaluer le temps des 4 fonctions du schéma de Regev $\lambda = 32$.

2. Evaluer le temps d'exécution moyen pour un nombre pertinent d'entrée uniformément tirés, pour chacune de ces fonctions.
3. Construire un jeu permettant d'évaluer le temps des 4 fonctions du schéma de Dual-Regev $\lambda = 32$.
4. Evaluer le temps d'exécution moyen pour un nombre pertinent d'entrée uniformément tirés, pour chacune de ces fonctions.
5. Calculer la taille de la clé publique, de la clé privée ainsi que du chiffré, ainsi que des paramètres globaux nécessaires.
6. Présenter, comparer et expliquer vos résultats.

(SecuriteFR) **Exercice 7.**

Analyse de la sécurité des deux schémas

1. Rappeler la définition de IND-CCA(1 | 2).
2. Les schémas Regev et Dual-Regev ne sont pas IND-CCA2, montrer le par une attaque pour $\lambda = 32$.
3. Essayer d'attaquer vos schémas de différentes manières. (timing attacks, Chosen-Ciphertext attacks, ...)