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 $\sigma = 10^3, m = 10^3$

aussienneDiscreteFR) **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*

Le premier schéma à implémenter est le schéma de chiffrement proposé par Regev en tant que premier cas d'usage de l'hypothèse LWE.
Oded Regev, « On lattices, learning with errors, random linear codes, and cryptography », In Proc. of STOCS, 2005
Afin d'assurer l'optimalité des paramètres, on choisira le plus petit paramètre satisfaisant chacune des conditions.

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

   On dénote ici, par $n = \lambda$ la taille du secret LWE $\mathbf{s}$ On considère en tant que paramètres globaux, les différentes variables précisées ci-dessous validant les conditions suivantes:

   - le modulo q est un nombre entier premier,
   - m est le nombre d'échantillons LWE satisfaisant $m \geqslant 4(n+1)\log q$,
   - l'écart type de la distribution gaussienne discrète de l'erreur est $\sigma := \alpha q \in \left]2\sqrt{n}, \frac{q}{4m}\right[$,
   - $q > 8m\sqrt{n}$ assurant la difficulté de LWE.

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

2. Implémenter la fonction **RegevKeyGen** générant à la fois une clé publique ansi qu'une clé privée associée. On rappelle la définition de la fonction:

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

3. Implémenter la fonction **RegevEncrypt** qui chiffre un bit b en utilisant la clé publique pk, on rappelle la fonction:

   1. Echantillonner m bits uniformément et dénoter ce message $\mathbf{r}$,
   2. Construire le chiffré $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) := \left(\mathbf{r}^\top \mathsf{pk}_1, \mathbf{r}^\top \mathsf{pk}_2 + \left\lfloor \frac{q}{2} \right\rfloor \cdot b\right)$.

4. Implémenter la fonction **RegevDecrypt** qui, en recevant un chiffré $\mathbf{c} = (\mathbf{c}_1, c_2)$ ressort 0 si $c_2 - \mathbf{c}_1^\top \mathsf{sk}$ est plus proche de 0 que de $\left\lfloor \frac{q}{2} \right\rfloor$ et 1 sinon.

5. Tester l'exactitude de vos fonctions i.e. que le déchiffrement d'un chiffré donne bien le bon clair choisi initial pour $\lambda = 32, 64, 128$.

6. Etendre les définitions de chiffrement et déchiffrement à un ensemble de bit (par concaténation).

(DualRegevFR) **Exercice 5.** *Chiffrement Dual-Regev*

Le chiffrement Dual-Regev est une autre proposition d'un schéma de chiffrement basé sur LWE par Gentry Peikert et Vaikutanathan. 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.
C. Gentry, C. Peikert, and V. Vaikuntanathan."Trapdoors for hard lattices and new cryptographic constructions." In Proc. of STOC, pages 197–206. ACM, 2008
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 et construit une variable globale **params** comprenant les différentes variables publiques fixées avant la génération des clés. On dénote ici, par $n = \lambda$ la taille du secret LWE $\mathbf{s}$, qui **ne sera pas** la clé privée. On considère en paramètres globaux les différents paramètres précisés ci-dessous et validant les conditions suivantes:

   - le modulo q est un nombre entier premier,
   - m est le nombre d'échantillons LWE satisfaisant $m \geqslant 4n \log q$,
   - l'écart type de la distribution gaussienne discrète de l'erreur est $\sigma := \alpha q \in \left]2\sqrt{n}, \frac{q}{8m}\right[$,
   - $q > 8m\sqrt{n}$ assurant la difficulté de LWE.

   On considère également en tant que paramètre global la matrice $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$.
   Afin d'assurer l'optimalité, on choisira les plus petits paramètres satisfaisant chacune des conditions.

2. Implémenter la fonction **DualKeyGen** générant à la fois une clé publique ansi qu'une clé privée associée. La clé privée sk est un vecteur m bits uniformément tiré. La clé publique associée pk est définie par $\mathsf{pk} = \mathsf{sk}^\top \mathbf{A}$.

3. Implémenter la fonction **DualEncrypt** qui chiffre un bit b en utilisant la clé publique pk, on rappelle la fonction:

   1. Construire un tuple $\mathsf{LWE}_{n,m,q,\sigma}$, dénoté $\mathbf{b}$ avec les élements secrets $\mathbf{s}$ et $\mathbf{e}$. En reprenant la définition de LWE, dans quel ensemble doivent-il être échantillonnés ?
   2. Construire le chiffré $\mathbf{c} = (\mathbf{c}_1, c_2) := \left(\mathbf{b}, \mathsf{pk}^\top \mathbf{s} + e' + \lfloor \frac{q}{2} \rfloor \cdot b\right)$ avec une erreur additionnelle $e' \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}$.

4. Implémenter la fonction **DualDecrypt** qui, en recevant, un chiffré $\mathbf{c} = (\mathbf{c}_1, c_2)$ ressort 0 si $c_2 - \mathbf{c}_1^\top \mathsf{sk}$ est plus proche de 0 que de $\lfloor \frac{q}{2} \rfloor$ et 1 sinon.

5. Tester l'exactitude de vos fonctions i.e. que le déchiffrement d'un chiffré donne bien le bon clair choisi initial pour $\lambda = 32, 64, 128$.

6. Etendre les définitions de chiffrement et déchiffrement à un ensemble de bit (par concaténation).

(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, ...)

3 Aller plus loin: versatilité de Dual-Regev

(UPKEDualRegevFR) **Exercice 8.** *Schéma de chiffrement actualisable*

1. Le schéma Dual-Regev a la particularité d'être très versatile et modifiable, permettant de construire de nouveaux schémas cryptographiques avancées. Donner quelques exemples de primitives cryptographiques avancées ainsi que des schémas associés, basés sur Dual-Regev.

2. Rechercher la définition de Schéma de chiffrement Actualisable (Updatable Public Key Encryption en anglais).

3. Basé sur ce papier, construire le schéma Dual-Regev associée basé sur une version modifiée de LWE (demander à votre enseignant si vous souhaitez davantage d'informations). Quelques fonctions (**SampleCode** et les fonctions qu'elle appelle) sont à obtenir du code originelle disponible ici, avec la documentation associée. Attention le Dual-Regev utilisé est modifié (notamment les ensembles d'échantillonage) voir la Definition 4 du papier (considérer le chiffrement d'un seul bit i.e $p = 2$).

4. Construire les deux fonctions d'actualisation de l'UPKE: **UpdPK** et **UpdSK** comme le détaille la Figure 4 du papier.

5. Afin que votre implémentation soit correct, rechercher les différentes conditions que doivent satisfaire les paramètres globaux q, n, m, σ. Tester votre implémentation pour $\lambda = 32, 64, 128$.