# Cryptographie post quantique : TP4 - Cryptanalyse de schéma de réseaux

Après avoir implémentée la variante structurée de schémas de KEM et de signatures, il est essentiel de connaitre la sécurité de vos schémas. Précédemment, les paramètres choisies en fonction du paramètre de sécurité en entrée vous étaient initialement définies. Dans un premier temps, vous allez découvrir la cryptanalyse sur les réseaux euclidiens. Les réductions de sécurité montrent que LWE/SIS sont au moins aussi difficile que des problèmes de réseaux, mais il est assez facile de voir que par la structure si l'on sait résoudre les problèmes fondamentaux des réseaux alors on arrive à résoudre SIS et LWE.
Ensuite, vous utiliserez un outil appelé lwe-estimator, permettant d'estimer la difficulté d'un problème LWE en fonction des paramètres en entrée. L'outil ne ressort pas des paramètres mais bien un paramètre de sécurité provenant de plusieurs estimations d'attaques sur LWE. Cet outil est (très) utilisé en pratique afin d'assurer un choix de paramètres judicieux et efficaces.

## 0.1 Paquetages nécessaires

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

- LWE
- Echantillonneur uniforme
- LWE Estimator
- Générateur de mauvaises bases

# 1 Cryptanalyse sur les réseaux

sage propose nativement des fonction de réduction de réseaux (LLL et BKZ) ainsi que des implémentations d'algorithme résolvant les problèmes fondamenteux CVP et SVP.

(AttaqueSVPetCVPFR) **Exercice 1.** *Résoudre SVP et CVP*

1. Générer une base d'un réseau difficile grâce à la commande **gen_lattice**. Vous définirez comme paramètres d'entrée $m = \dim$ et $n = \frac{dim}{2}$. Votre objectif est de faire varier la valeur du modulo $q$ afin d'étudier la vitesse de résolution de SVP dans le réseau engendré.
   - Soit $q = \mathcal{O}(\log(n)), q = \mathcal{O}(n)$ ou $q = \mathcal{O}(n^2)$,
   - Où $q$ est soit premier, soit une puissance de 2.

   Tester pour des dimensions $\dim = 38, 40, 42$. Qu'en déduisez-vous ? (Vous pouvez la méthode **shortest_vector** sur un objet **IntegerLattice**).

2. Avec les mêmes paramètres que précédemment, résoudre le problème du vecteur le plus proche avec la méthode **closest_vector**.
   - Pour une cible $\mathbf{u} = \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} \in \mathbb{Z}_q^m$,
   - Pour une cible $\mathbf{u} \in \Lambda$, le réseau engendré précédemment,
   - Pour une cible aléatoire $\mathbf{u} \notin \Lambda$ (Vérifier bien qu'il n'appartient pas au réseau avant la résolution).

(AttaqueSISFR) **Exercice 2.** *Résoudre SIS*

Vous avez vu dans votre cours que le problème SIS était au moins aussi difficile que résoudre GapSVP. L'objectif est d'utiliser le solveur **shortest_vector** de l'exercice précédent afin de trouver un court vecteur solution d'un problème SIS.

1. Générer une matrice aléatoire associé à un problème SIS en utilisant **random_matrix**. Les dimensions de cette matrices seront $m = 20$ et $n = 10$, et on spécifiera un modulo $q = 127$.

2. On peut utiliser la méthode **shortest_vector** sur des objets **IntegerLattice**. Avant ça, trouvez une base du réseau engendré et associé à la matrice précédente, plusieurs indices:

   - le problème SIS revient à trouver un petit vecteur dans le réseau orthogonal $\Lambda_q^{\perp}(\mathbf{A})$,
   - on peut en trouver une base de $\Lambda^{\perp}(\mathbf{A}) = \mathsf{Ker}(\mathbf{A})$ avec **.right_kernel_matrix()**,
   - Ici, on souhaite les vecteurs orthogonaux dans le réseau modulo $q$. C'est-à-dire qu'on se place dans un réseau modulo $q$ où les vecteurs de la forme $\begin{bmatrix}0 & \cdots & 0 & q & 0 & \cdots & 0\end{bmatrix}$ sont des vecteurs de la base du réseau. On peut ainsi construire de nouveaux vecteurs linéairement combinés avec ceux de $\Lambda^{\perp}(\mathbf{A})$.

3. Vérifier que la norme (euclidienne) du vecteur est bien petite. (On centrera d'abord le vecteur en o si ce n'est pas déjà fait). Vous utiliserez la méthode **.norm(2)** afin de calculer la norme d'un vecteur.

(AttaqueLWEFR) **Exercice 3.** *Résoudre LWE*

L'objectif maintenant est de montrer l'analogie de ces attaques au problème LWE pour des samples $(\mathbf{A}, \mathbf{As} + \mathbf{e})$. En ayant une erreur $\mathbf{e}$ petite, vous avez vu que le problème (search) LWE revient en réalité à résoudre un problème CVP pour un réseau $\Lambda_q(\mathbf{A})$ de dimension $m$ (le nombre d'échantillons) et de rang $n$ (la taille de $\mathbf{s}$).

1. Générer un problème LWE de petite dimension grâce à la fonction **lwe** en considérant $q = 127, n = 4, m = 5$ et où le secret sera un vecteur avec éléments tirés uniformément dans $\{-1, 1\}^n$.

2. Trouvez une matrice génératrice associée au réseau engendré par la matrice $\mathbf{A}$ associé au problème CVP sous-jacent.

3. Résoudre un problème CVP pour un bon vecteur cible permettant de retrouver le vecteur $\mathbf{s}$ de l'échantillon LWE précédent.

   **Aide:** Aidez-vous des méthodes **closest_vector()** et **solve_right()**. Pensez à la construction du réseau de la section précédente. Attention également à quelle classe appartiennent vos objets, ils doivent être recast soit en **matrix** soit en **vector** soit en **IntegerLattice** (ou autre encore).

Vous savez que les problèmes de réseaux sont faciles quand la dimension est petite, ainsi nous nous sommes placés en petites dimensions afin de pouvoir exécuter les méthodes de résolution de problèmes. Cependant, en petite dimension il peut y avoir des collisions en terme de solution et donc il est possible que vous trouviez un autre vecteur solution $\mathbf{s}'$ que celui utilisée localement par l'objet LWE (accessible avec la méthode **_LWE_s**).

## 2 LWE Estimator

Martin R. Albrecht, Rachel Player and Sam Scott. On the concrete hardness of Learning with Errors. Journal of Mathematical Cryptology. Volume 9, Issue 3, Pages 169–203, ISSN (Online) 1862-2984, ISSN (Print) 1862-2976 DOI: 10.1515/jmc-2015-0016, October 2015

Le LWE-Estimator est un outil fréquemment utilisé afin de s'assurer d'avoir des paramètres permettant la sécurité des constructions cryptographiques basées sur les réseaux et plus spécifiquement LWE. Importez l'estimateur et plus particulièrement les fichiers du dossier *estimator*.

(InitLWEEstimatorFR) **Exercice 4.** *Initiation au LWE Estimator*

Voyons dans un premier temps, la construction des différents objets de cette librairie.

1. Dans **schemes**, vous trouverez des jeux de paramètres déjà bien établis. Voici un exemple, celui de la semaine précédente, le KEM Kyber pour un paramètre de sécurité $\lambda = 128$. Le problème LWE sur lequel on assure la sécurité devra donc prendre les valeurs suivantes. On identifie 5 paramètres différents:

   - $n$: la taille du secret,
   - $m$: le nombre de sample LWE,
   - $q$: le modulo utilisé,
   - $Xs$: la distribution du secret,
   - $Xe$: la distribution de l'erreur.

   **Par exemple**: schemes.Kyber512 est un objet **LWEParameters** avec les paramètres de Kyber512 (ancien nom de ML-KEM pour $\lambda = 128$).

   Ici $\mathsf{D}(\sigma = 1.22)$ signifie Distribution Gaussienne d'écart-type $\sigma = 1.22$,

Une fois les paramètres choisies dans un objet LWEParameters qu'on nommera paramsLWE, on peut exécuter la fonction LWE.estimate(paramsLWE).

2. La fonction LWE.estimate estime la difficulté de plusieurs attaques optimisées de LWE selon vos paramètres en entrée, afin de déterminer la facilité (ou non) de performer ces attaques sur des problèmes LWE avec les paramètres instanciés. Les différentes attaques estimées sont:

   - BKW,
   - $\mu$SVP,
   - BDD,
   - Attaque "dual",
   - Attaque "dual_hybrid",

   Exécuter estimation = LWE.estimate(schemes.Kyber512) pour en avoir un aperçu. Puis essayer sur d'autres schémas présent dans **schemes**.

   **Aide:** L'entrée importante a regardé est l'entrée "**rop**" signifiant "ring operations" c'est-à-dire le nombre d'opérations nécessaire à chaque attaque afin de "casser" le schéma. Chaque attaque a des coûts différents et il faut vérifier que l'attaque la plus rapide a besoin de réaliser au moins $2^\lambda$ operations (minimum) où $\lambda$ est le paramètres de sécurité.

3. Analyser la robustesse de votre jeu de paramètres en instanciant un nouvel objet LWE.parameters et exécuter LWE.estimate sur cet objet.

4. Trouvez un jeu de paramètres satisfiant $\lambda = 64$ mais pas $\lambda = 65$.

(LWEEstimatorFR) **Exercice 5.** *Utilisation concrète du LWE Estimator*

Votre objectif est de manipuler le LWE-estimator afin d'avoir un jeu de paramètres assurant la sécurité pour un paramètres de sécurité en entrée $\lambda$.
Pour cela, faites un script permettant d'évaluer la difficulté des attaques sur les jeux de paramètres en entrée, l'objectif est de faire évoluer vos jeux de paramètres jusqu'à atteindre un point fixe.

1. Vous devez construire un jeu de paramètre pour une implémentation, pour laquelle sa sécurité serait basée sur une instance d'un problème LWE pour laquelle:

   - le paramètre de sécurité souhaité est $\lambda = 64$ (et ne doit pas satisfaire $\lambda = 65$).
   - $q$ est un nombre premier de la forme $q = 2^k + 1$ et $q = \omega(n)$,
   - le vecteur **s** et le vecteur **e** suivent chacun une loi uniforme avec coefficients appartenenant à $[-1, 0, 1]$,
   - la matrice **A** sera une matrice carré à coefficients dans $\mathbb{Z}_q^{m \times n}$ où $m = n$.

   **Aide :** En utilisant une dichotomie, wous allez procéder de la manière suivante:

   1. Faites évoluer $k$ pour avoir $q = 2^k + 1$ étant premier,
   2. Considérer $n = m = q$ et rechercher un $q$ valide tel que le jeu valide une bonne sécurité,
   3. Une fois que vous avez un bon ensemble de paramètres, refaites un script recherchant $n = m$ le plus judicieux possible avec une dichotomie (c'est-à-dire le plus petit assurant tout de même la sécurité).

2. Vous devez construire un jeu de paramètre pour une implémentation, pour laquelle sa sécurité serait basée sur une instance d'un problème MLWE pour laquelle:

   - le paramètre de sécurité souhaité est $\lambda = 64$ (et ne doit pas satisfaire $\lambda = 65$).
   - $d$ sera le degré du polynôme engendrant l'idéal $< X^d + 1 >$, $d$ est une puissance de 2 (plus il est grand mieux c'est),
   - le vecteur **s** et le vecteur **e** suivent chacun une loi uniforme dans à coefficient $[-1, 0, 1]$,
   - rappel: $\mathcal{R}_q := \mathbb{Z}_q[X] / < X^d + 1 >$,
   - la matrice **A** sera une matrice carré à coefficients dans $\mathcal{R}_q^{m \times n}$ où $m = n$.

   **Aide:** On analyse la sécurité d'un problème MLWE de la même manière qu'un problème LWE (mais pour des tailles $md \times nd$). En effet, on suppose que la structure n'induit pas de faiblesse lors des attaques, ainsi on ne considèrera pas la structure dans la recherche des paramètres optimaux, simplement les dimensions.

3. **(Facultatif)** Modifier les questions précédentes afin d'obtenir des jeux de paramètres loptimisés pour $\lambda = 128, 196, 256$.