

Cryptographie post quantique : TP2 - Sécurité de Regev et LWE Estimator

Après avoir implémenté un schéma cryptographique, il est essentiel de connaître 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, nous allons redécouvrir la sécurité prouvée en reliant la difficulté de résolution des problèmes de réseaux à celle du schéma.

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

Récupérer votre implémentation du chiffrement de Regev du TP1.

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 Sécurité du chiffrement de Regev

(SecuriteRegevFR) Exercice 1.

Sécurité de Regev

1. Rappeler le schéma de chiffrement de Regev et ses 4 fonctions principales.
2. Quelle est la condition pour s'assurer que le schéma soit correct ?
3. Rappeler les variantes calculatoire et décisionnelle du problème Learning With Errors (LWE).
4. Montrer que la distribution de la clé publique est calculatoirement indistinguishable de la distribution uniforme sur $\mathbb{Z}_q^n$. Pourquoi est-ce important pour la sécurité de ce schéma ?
5. Rappeler la définition de la sécurité IND-CPA (sous forme d'expérience).
6. Montrer que le schéma de chiffrement de Regev est IND-CPA sous l'hypothèse que le problème LWE est difficile. On le note $\text{LWE} \implies \text{IND-CPA Regev}$.

Aide: Utiliser le left-over-hash lemma pour l'étude de la distribution $\mathbf{c}_0$ et $\mathbf{c}_1|_{\text{msg}}$.

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 2.**

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,
- X_s : la distribution du secret,
- X_e : 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 $D(\sigma = 1.22)$ 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,
 - μ 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^λ operations (minimum) où λ est le paramètre de sécurité.

3. Analyser la robustesse de votre jeu de paramètres du TP1 en instanciant un nouvel objet `LWE.parameters` et en exécutant `LWE.estimate` sur cet objet.

4. Trouvez un jeu de paramètres satisfaisant $\lambda = 64$ mais pas $\lambda = 65$. Modifiez les paramètres de **DualGenerator** en conséquence et testez votre implémentation.

Aide: Faites par dichotomie.

5. Trouvez le jeu de paramètres optimale satisfaisant $\lambda = 64$ tout en garantissant des tailles minimales (clés et/ou chiffrés). Modifiez les paramètres de **RegevGenerator** en conséquence et testez votre implémentation.

Aide: Faites par dichotomie.

3 Facultatif : s'entraîner avec le LWE Estimator

(LWEEstimatorFR) **Exercice 3.**

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 λ .

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 $\mathbf{s}$ et le vecteur $\mathbf{e}$ suivent chacun une loi uniforme avec coefficients appartenant à $[-1, 0, 1]$,
- la matrice $\mathbf{A}$ sera une matrice carré à coefficients dans $\mathbb{Z}_q^{m \times n}$ où $m = n$.

Aide : En utilisant une dichotomie, vous 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 $\langle X^d + 1 \rangle$, d est une puissance de 2 (plus il est grand mieux c'est),
- le vecteur $\mathbf{s}$ et le vecteur $\mathbf{e}$ suivent chacun une loi uniforme dans à coefficient $[-1, 0, 1]$,
- rappel: $\mathcal{R}_q := \mathbb{Z}_q[X] / \langle X^d + 1 \rangle$,
- la matrice $\mathbf{A}$ 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 considérera 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 optimisés pour $\lambda = 128, 196, 256$.