MPI – Projet d'informatique n°2
Post-quantum cryptography : Learning With Errors

Florian Bourse

À rendre pour le mercredi 19 octobre

Le but de ce projet est d'implémenter un schéma de chiffrement à clé publique post-quantique, c'est-à-dire qui protège la confidentialité des données même contre un attaquant possédant un ordinateur quantique. Cette construction est due à Oded Regev en 2005.

1 Learning With Errors

La sécurité du système de chiffrement que nous allons implémenter repose sur la difficulté du problème Learning With Errors (LWE), que l'on peut résumer de la manière suivante :
Soit $\mathbf{s}$ un vecteur uniformément aléatoire d'entiers modulo un certain nombre q.

$$(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$$

est indistinguable d'un élément choisi uniformément aléatoirement, pour $\mathbf{a}$ un vecteur uniformément aléatoire, et e qui provient d'une « distribution d'erreur ».
Pour notre implémentation nous allons utiliser les paramètres suivants :

- $q = 2^{19} - 1$, tous les calculs seront effectués modulo q ;
- $n = 1000$, la dimension des vecteurs $\mathbf{s}$ et $\mathbf{a}$;
- $\mathcal{D}_\sigma$, la distribution d'erreur est une distribution Gaussienne discrète de paramètre σ ;
- $\sigma = 1.$, le paramètre de la distribution de l'erreur e.

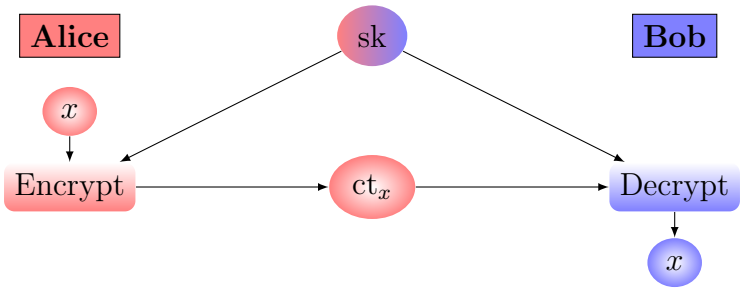
2 Distribution Gaussienne discrète

On note $\rho(x) = e^{-\frac{x^2}{2\sigma^2}}$ la fonction Gaussienne de paramètre σ.
La distribution Gaussienne discrète de paramètre σ assigne à chaque entier x une probabilité proportionnelle à $\rho(x)$.
Cette distribution est centrée autour de 0, la probabilité qu'elle dépasse une certaine borne peut être estimée en utilisant la fonction `erf`, sur `www.wolframalpha.com` par exemple.
Ici, on supposera que les entiers en dehors de $\{-16; -15; \ldots; 14; 15\}$ ne sont jamais tirés.
On aura donc pour $X \leftarrow \mathcal{D}_\sigma$,

$$P(X = x) = \frac{\rho(x)}{\sum_{k=-16}^{1} 5\rho(k)}$$

Pour tirer un élément de cette distribution, on range dans un tableau les $P(X \leq i)$ puis on tire un nombre aléatoire ε entre 0 et 1 et on choisit comme valeur le premier nombre i tel que $P(X \leq i) \geq \varepsilon$.

FIGURE 1 – Chiffrement symétrique



3 Chiffrement symétrique

Le principe de fonctionnement d'un schéma de chiffrement symétrique est décrit sur la figure 1. Un tel outil permet à deux personnes partageant une clé secrète de communiquer de manière sécurisée. C'est sur ce principe que repose la sécurité des communications sur internet par exemple.

Pour construire un schéma de chiffrement symétrique dont la sécurité repose sur la difficulté de LWE, la clé secrète est $\mathbf{s}$, et pour chiffrer un message μ, on l'encode par $v = \lfloor \mu \frac{q}{M} \rceil$, où M est le nombre de messages possibles, et on donne le chiffré $(\mathbf{a}, b)$, où $b = \langle \mathbf{a}, \mathbf{s} \rangle + e + v$, avec M la taille de l'espace des messages. Nous allons chiffrer des caractères, donc $M = 256$.
Pour déchiffrer $(\mathbf{a}, b)$ avec la clé $\mathbf{s}$, on calcule $v = b - \langle \mathbf{a}, \mathbf{s} \rangle$ et on décode $\mu = \lfloor v \frac{M}{q} \rceil$.
Le résultat est correct tant que l'erreur e est plus petite que $\frac{q}{2M}$.

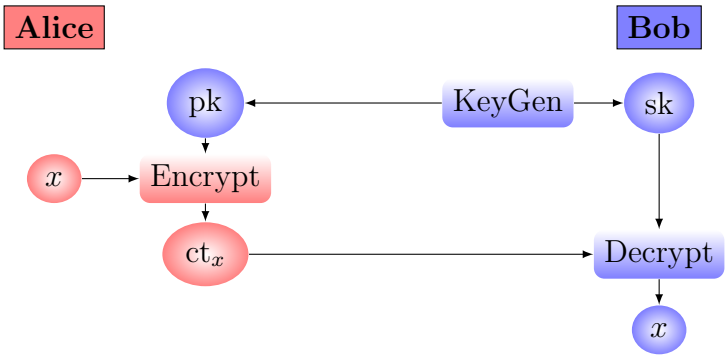
L'inconvénient de ce type de schéma de chiffrements est la nécessité de partager une clé secrète avant l'échange. Cependant, ce schéma de chiffrement est malléable, c'est-à-dire que l'on peut changer le contenu du chiffré sans avoir la clé secrète.

Question 1. Proposer une fonction `twist` qui permet de décaler de i le caractère chiffré.
Par exemple, décaler de 2 le caractère 'a' doit donner le caractère 'c'.

Alice et Bob peuvent sécuriser leur communication de la manière suivante :

1. Alice construit un chiffré `ct` contenant un nombre aléatoire k et l'envoie à Bob ;
2. Bob twiste `ct` en ajoutant son message μ et le renvoie à Alice ;
3. Alice déchiffre et obtient $k + \mu$ et n'a plus qu'à enlever k pour obtenir le message.

FIGURE 2 – Chiffrement asymétrique



4 Chiffrement asymétrique

Le protocole présenté dans la section précédent ne permet pas à Alice d'être sur que le message provient de Bob. Pour résoudre ce problème, on peut utiliser des signatures électroniques, ou alors utiliser un schéma de chiffrement à clé publique, ou asymétrique, dont le fonctionnement est décrit dans la figure 2.

Pour construire un schéma de chiffrement à clé publique dont la sécurité repose sur la difficulté de LWE, on réutilise le schéma de chiffrement à clé secrète précédent, et on ajoute une clé publique, qui est composée de m chiffrés de 0

$$\mathtt{pk}_i = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i).$$

Dans notre cas, on prendra $m = 19000$.

Pour chiffrer un message μ en utilisant la clé publique, on fait la somme d'un sous-ensemble aléatoire de ces chiffrés de 0, puis on ajoute la valeur v qui code le message μ.

$$\mathtt{ct}_\mu = \sum_{i=1}^{m} r_i \mathtt{pk}_i + (\mathbf{0}, \lfloor \mu \frac{q}{M} \rceil),$$

où r_i est soit 0, soit 1, avec probabilité $\frac{1}{2}$.

5 Spécifications techniques

À la fin, votre programme doit avoir 3 comportements différents, qui seront appelés grâce à un argument en ligne de commande.

`keygen :` Génère une paire de clé privée/clé publique associées, et les écrits respectivement dans les fichiers `.ssh/lwe` et `.ssh/lwe.pub` du home par défaut. Le nom du fichier de la clé secrète peut être changé en étant donné en ligne de commande, celui de la clé publique possède le même nom avec `.pub` à la fin. Si ces fichiers existent déjà, ne fait rien et renvoie une erreur.

`encrypt :` Lit la clé publique dans le fichier `.ssh/lwe.pub` du home par défaut, puis lit un par un les caractères sur l'entrée standard, et écrit sur la sortie standard le chiffré correspondant à ce caractère, ce jusque la fin du fichier (ctrl + D pour terminer l'entrée standard). L'emplacement de la clé publique peut être changée en étant donné en ligne de commande.

`decrypt :` Lit la clé secrète dans le fichier `.ssh/lwe` du home par défaut, puis lit un par un les chiffrés sur l'entrée standard, et écrit sur la sortie standard le caractère correspondant après l'avoir déchiffré, ce jusque la fin du fichier. L'emplacement de la clé secrète peut être changée en étant donné en ligne de commande.

Par exemple, si l'exécutable après compilation s'appelle `lwe`.
Après avoir créer une paire de clé :

```
./lwe keygen key
```

La commande suivante permet de créer un fichier `ciphertext.txt` dont le contenu est celui du fichier `message.txt` qui a été chiffré.

```
./lwe encrypt key.pub < message.txt > ciphertext.txt
```

Le message original peut ensuite être retrouvé avec

```
./lwe decrypt key < ciphertext.txt > message.txt
```

On note que la commande suivante doit se comporter comme `echo` :

```
./lwe encrypt key.pub | ./lwe decrypt key
```