Arithmétique et DLP

Version du 2 mars 2019

TME

Exercice 1 – Arithmétique de base

1. Écrire un programme permettant de calculer un PGCD et une relation de Bézout entre deux entiers.
2. Écrire une fonction permettant de calculer des inverses modulaires. En déduire une fonction permettant de calculer tous les inversibles de l'anneau $\mathbb{Z}/N\mathbb{Z}$ pour N un entier donné.
3. On rappelle que l'indicateur d'Euler d'un entier N, noté $\phi(N)$, représente le nombre d'entiers positifs et plus petit que N premier avec N. Écrire une fonction permettant de faire son calcul.

Exercice 2 – Logarithme discret

Dans cet exercice, on s'intéresse au problème du logarithme discret dans le cas du groupe cyclique $G = (\mathbb{Z}/p\mathbb{Z})^\times$ des inverses modulo un nombre premier p.

Instanciation du problème du logarithme discret

1. Implémenter la version itérative de l'exponentiation modulaire comme vu en Cours et en TD.
2. Écrire une fonction permettant de factoriser un nombre n. Cette fonction retournera la liste des couples (p, v_p) pour tous les facteurs premiers p avec v_p la valuation p-adique associée.
3. En utilisant la sortie de la fonction de la question précédente pour $p - 1$, en écrire une permettant de calculer l'ordre d'un élément $g \in G$.
4. À l'aide de la fonction précédente, implémenter une fonction permettant de trouver un générateur du groupe cyclique G.

À partir de ces fonctions il est possible de trouver un groupe G et un de ses générateurs pour créer des problèmes de logarithme discret. Cependant, en cryptographie, il est nécessaire que p soit un très grand nombre premier et la factorisation de $p - 1$ risque alors d'être très coûteuse. Afin d'éviter l'utilisation de la méthode de Pohlig-Hellman, il est également nécessaire que $p - 1$ ait un minimum de facteur. Nous cherchons donc à générer p de la forme $p = 2q + 1$ où q est un entier premier, p est alors appelé nombre premier sûr.

5. En utilisant la fonction de test de primalié `is_probable_prime` (disponible sur gitlab), écrire une fonction qui génère un nombre premier sûr en fonction d'une longueur donnée en nombre de bits.

Baby-Step Giant-Step

6. Implèmenter l'algorithme BSGS de Shanks pour la résolution du logarithme discret dans le groupe G. Vérifier l'algorithme avec un groupe dont l'ordre est un nombre premier p de 32 bits.

Algorithme ρ de Pollard

Nous sommes toujours dans $G = (\mathbb{Z}/p\mathbb{Z})^\times$ où p est un nombre premier mais nous nous intéressons dorénavant au logarithme discret dans le sous-groupe $\langle h \rangle = \{1, h, h^2, h^3, \dots\}$ engendré par h d'ordre premier q. Par exemple si $p = 2q$ est un nombre premier sûr et g est un générateur de G, alors $h = g^2$ est d'ordre q qui est premier.

7. Soit n un élément de $\langle h \rangle$, nous cherchons à calculer son logarithme en base h.
   L'idée clé de l'algorithme rho de Pollard est de trouver des entiers a, b, A, B tels qu'on ait la relation
   $$h^a \cdot n^b = h^A \cdot n^B \,. \tag{1}$$
   On suppose que $(B - b)$ est premier avec q.
   Vérifier que l'on peut alors retrouver le logarithme discret de n à l'aide de la formule
   $$(a - A)(B - b)^{-1} \mod q \,.$$

Pour trouver une relation du même type que (1), l'algorithme utilise une fonction $f\colon \langle h \rangle \to \langle h \rangle$ pseudo-aléatoire et utilise la suite $(x_i)_{i \in \mathbb{N}}$ où $x_0 = h^{a_0} n^{b_0}$ est choisi en prenant a_0 et b_0 aléatoires et pour tout $i > 0$, $x_i = f(x_{i-1})$.
On définit
$$f(x) = \begin{cases} hx & \text{si } x = 0 \mod 3 \\ nx & \text{si } x = 1 \mod 3 \\ x^2 & \text{si } x = 2 \mod 3 \,. \end{cases}$$
Une telle fonction est à la fois simple à évaluer et suffisamment aléatoire mais elle permet surtout de toujours connaître une décomposition de x_i en fonction de h et de n :
$$x_i = h^{a_i} n^{b_i} \,.$$
On a en effet les relations suivantes pour tout $i > 0$:
$$a_i = \begin{cases} a_{i-1} + 1 \mod p & \text{si } x_{i-1} = 0 \mod 3 \\ a_{i-1} \mod p & \text{si } x_{i-1} = 1 \mod 3 \\ 2 \cdot a_{i-1} \mod p & \text{si } x_{i-1} = 2 \mod 3 \,, \end{cases} \qquad b_i = \begin{cases} b_{i-1} \mod p & \text{si } x_{i-1} = 0 \mod 3 \\ b_{i-1} + 1 \mod p & \text{si } x_{i-1} = 1 \mod 3 \\ 2 \cdot b_{i-1} \mod p & \text{si } x_{i-1} = 2 \mod 3 \,. \end{cases}$$

8. Écrire une fonction `next` qui à partir de (x_i, a_i, b_i) retourne $(x_{i+1}, a_{i+1}, b_{i+1})$.

En itérant l'application des fonctions `next` et `next` $\circ$ `next`, on peut calculer en parallèle les suites $(x_i)_{i \in \mathbb{N}}$ et $(x_{2i})_{i \in \mathbb{N}}$ ainsi que les exposants correspondant $(a_i)_{i \in \mathbb{N}}$, $(b_i)_{i \in \mathbb{N}}$, $(a_{2i})_{i \in \mathbb{N}}$ et $(b_{2i})_{i \in \mathbb{N}}$.
La théorie sur les fonctions aléatoires nous dit alors qu'il existe un k tel que
$$h^{a_k} n^{b_k} = x_k = x_{2k} = h^{a_{2k}} n^{b_{2k}} \,,$$
avec, en moyenne, k de l'ordre de $\mathcal{O}(\sqrt{q})$.

9. En utlisant ce principe et la réponse à la question 7, implémenter l'algorithme ρ de Pollard résolvant le problème du logarithme discret dans le groupe $\langle h \rangle$. (Dans le cas où $B - b$ n'est pas inversible, une solution est de recommencer avec un nouveau x_0 pris aléatoirement.)
10. Donner la complexité de cet algorithme. Quel est l'avantage de cet algorithme par rapport à Baby-Step Giant-Step ?