Arithmétique et DLP

Version du 3 mars 2024

TME

1 Arithmétique de base

Exercice 1 – L'algorithme d'Euclide étendu, outil incontournable de l'arithmétique modulaire

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.

2 Logarithme discret

Dans cette partie, 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.

Exercice 2 – 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 l'exposant de p dans n.
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.

Exercice 3 – Nombres premiers sûrs

À partir des fonctions précédentes 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 le moins de facteurs possibles. 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*.

En utilisant la fonction de test de primalité `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.

Exercice 4 – Baby-Step Giant-Step

1. 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.