Introduction à la cryptographie
TD7 – Echange de clefs

Cécile Pierrot

25 février 2020

1 Echange de clefs par le protocole de Diffie et Hellman

Supposons que Alice souhaite construire une clef secrète avec vous. Elle vous a envoyé les valeurs suivantes :

— $g = 13$
— $p = 97$
— $g^a = 42$

Question 1. De quel ensemble g doit-il être un générateur ? Ecrivez un petit programme qui vérifie que c'est bien le cas.

Supposons maintenant que vous choisissiez un entier secret $b = 5$.

Question 2. Quelle information devez-vous envoyer ?

Question 3. Quelle est votre clef commune ?

2 Calcul de logarithme discret par pas de bébés et pas de géants

Etant donné un groupe multiplicatif cyclique G et un générateur g, résoudre le problème du logarithme discret consiste, étant donné un élément arbitraire h du groupe, à retrouver un entier x tel que :

$$g^x = h.$$

On appelle logarithme discret de h en base g un tel entier x (qui n'est pas unique, mais déterminé modulo l'ordre du groupe). En cryptographie en pratique, ce problème est à la base de très nombreux protocoles asymétriques. Le groupe G peut être de nature différente :

— il peut s'agir du groupe (additif) des points sur une courbe elliptique
— ou bien du groupe des éléments inversibles sur un corps fini.

Pas de panique si vous ne connaissez aucun de ces deux types "d'objets" mathématiques. Nous allons prendre un cas particulier de corps fini : $\mathbb{Z}/p\mathbb{Z}$. Nous rappelons que $\mathbb{Z}/p\mathbb{Z}$ avec p un nombre premier est simplement l'ensemble des entiers de 0 à $p-1$. Tous les calculs se font modulo p. Dans ce cas, G est l'ensemble des éléments entre 1 et $p-1$, ce que l'on note $G = (\mathbb{Z}/p\mathbb{Z})^*$. On suppose que l'on connait un générateur g pour la multiplication. C'est-à-dire qu'il existe un entier g tel que $g, g^2, g^3, g^4, \cdots, g^{p-1}$ décrit exactement tous les entiers de 1 à $p-1$, modulo p.

Question 1. Montrez que 3 est un générateur de $(\mathbb{Z}/7\mathbb{Z})^*$.

Question 2. En vous aidant des calculs réalisés à la question précédente, déterminez quel est le logarithme discret de 5 en base 3 dans $(\mathbb{Z}/7\mathbb{Z})^*$. Est-il unique ?

Question 3. Proposez un algorithme naïf qui étant donné un modulo p premier et un générateur g de $(\mathbb{Z}/p\mathbb{Z})^*$, et un élément h, retourne un logarithme discret de h en base g. Implémentez le en une fonction logdinaif de Python par exemple. Parvenez-vous à résoudre le problème du logarithme discret pour l'exercice 1 (id est, pouvez-vous retrouver la clef secrète à partir des éléments qui sont publics ?).

Question 4. Même question avec $p = 4234249417, g = 1423318003$ et $h = 2159141874$.

L'algorithme de Shanks a été proposé en 1970 pour résoudre le problème du logarithme discret dans un groupe qui possède q éléments en $O(\sqrt{q})$ opérations, et en $O(\sqrt{q})$ en mémoire. Il s'appelle aussi l'algorithme "Pas de bébés, pas de géants". Cet algorithme est le plus performant lorsqu'on ne sait rien de plus sur G que le nombre d'éléments. Dans le cas des corps finis (et donc de $(\mathbb{Z}/p\mathbb{Z})^*$) il existe des algorithmes plus efficaces mais beaucoup beaucoup plus complexes, qu'on n'abordera pas ici.

Posons $T = \lceil \sqrt{q} \rceil + 1$. L'idée consiste à remarquer que l'égalité $h = g^x$ avec $x = x_1 T + x_0$ (avec $0 \leq x_1, x_0, < T$) s'écrit également :

$$h \cdot (g^{-T})^{x_1} = h g^{-x_1 T} = g^{x_0}.$$

Il suffit dans un premier temps de calculer les T valeurs prises par le terme de droite de cette égalité, c'est-à-dire de calculer la liste des éléments g^i pour i variant de 0 à $T-1$, et de stocker les valeurs obtenues (g^i, i). On parle de pas de bébés.

Dans un second temps l'algorithme calcule toutes les valeurs de gauche successivement, c'est-à-dire $h \cdot (g^{-T})^j$ pour j variant de 0 à $T-1$. Pour chacune d'entre elle elle compare si elle est déjà dans la liste pré-calculée. Comme les pas sont plus grands, on parle de pas de géants.

Question 5. Implémentez cet algorithme en Python pour le cas où $G = (\mathbb{Z}/p\mathbb{Z})^*$. Pouvez-vous maintenant trouver un logarithme discret pour $h = 2159141874$ en base $g = 1423318003$ modulo $p = 4234249417$?

Question 6. Combien de multiplications faut-il effectuer pour que cet algorithme trouve la solution dans le cas général (en fonction du nombre d'éléments q de G) ? Combien d'éléments sont stockés en mémoire.

```
def logdinaif(p,g,h):
    x=1
    gpuiss=g
    while not(gpuiss ==(h%p)):
        gpuiss=g*gpuiss %p
        x=x+1
    return x

def algodeshanks(p,g,h):
    T = int((p-1)**(1/2.0)) +1
    print T
    liste=[]
    x=1
    for i in range(1,T):
        x=x*g%p
        liste.append((i,x))
    print liste
    geant=h%p
```

```
print geant
for j in range(1,T):
    geant=(geant*((g**(-1)**T)%p))%p
    print geant
    for (i,x) in liste:
        if geant==x:
            return (i+Tj)%(p-1)
return False
```