École normale supérieure
Département d'informatique **2022-2023**

Introduction à la cryptologie
TD n° 1 : Logarithme Discret, ElGamal, Pedersen

Exercice 1 (Multi-exponentiation)**.** Soit $\mathbb{G}$ un groupe abélien (noté multiplicativement). Soient t éléments $g_1, \ldots, g_t$ du groupe $\mathbb{G}$ et des entiers positifs $n_1, \ldots, n_t < |\mathbb{G}|$. Proposer un algorithme qui calcule le produit $g_1^{n_1} \ldots g_t^{n_t} \in \mathbb{G}$ en $O(\ell + 2^t)$ multiplications dans $\mathbb{G}$ (où ℓ est la taille en bits de $\max(n_1, \ldots, n_t)$).

Exercice 2. Soit $\mathbb{G}$ un groupe d'ordre premier p généré par g. On rappelle la définition du chiffrement ElGamal dans $\mathbb{G}$.

— $\mathsf{KeyGen}(1^\lambda)$: tire $x \xleftarrow{U} \mathbb{Z}_p$ et renvoie la clé publique $\mathsf{pk} = g^x$ et la clé secrète $\mathsf{sk} = x$.

— $\mathsf{Enc}(\mathsf{pk}, m)$: pour $m \in \mathbb{G}$, tire $r \xleftarrow{U} \mathbb{Z}_p$ et renvoie $c = (m \cdot \mathsf{pk}^r, g^r)$.

On va discuter des propriétés de ElGamal, parfois dans le cadre des ≪ commitments ≫.

1. Comment peut-on déchiffrer un message chiffré c si on connaît sk ?
2. Peut-on déchiffrer sans connaître sk dans le cas où il est facile (resp. difficile) de calculer le logarithme discret dans $\mathbb{G}$?
3. Supposons que c'est dur de calculer g^{xy} étant donné (g, g^x, g^y). Montre que c'est dur de déchiffrer $c = (c_1, c_2)$ étant donné pk. Est-ce que cette propriété est suffisant pour un algortihme de chiffrement ?
4. Montre que ElGamal est homomorphe, c'est-à-dire : étant donné deux messages chiffrés $c = \mathsf{Enc}(\mathsf{pk}, m)$ et $c' = \mathsf{Enc}(\mathsf{pk}, m')$, il est possible de calculer c'' qui se déchiffre en $m'' = m \cdot m'$.
5. Supposons que les distributions (g, g^x, g^y, g^{xy}) et (g, g^x, g^y, g^c) sont difficiles à distinguer, où $x, y, c \xleftarrow{U} \mathbb{Z}_p$. Montre que ElGamal est ≪ cachant ≫ (*hiding*), c'est-à-dire : étant donné pk, pour tout choix de m_1, m_2, et pour $b \xleftarrow{U} \{0, 1\}$, il est difficile de décider si $c = \mathsf{Enc}(\mathsf{pk}, m_b)$ est un chiffrement de m_1 ou m_2.
6. Montre que ElGamal est ≪ contraignant ≫ (*binding*), c'est-à-dire : étant donné pk, il est difficile de trouver (c, m_1, m_2, r_1, r_2) tel que $\mathsf{Enc}(\mathsf{pk}, m_1; r_1) = \mathsf{Enc}(\mathsf{pk}, m_2; r_2)$ mais $m_1 \neq m_2$.
7. Si l'adversaire a un pouvoir calculatoire infini, ElGamal est-il toujours cachant et contraignant ?
8. Modifier ElGamal pour que l'espace de message soit $\mathbb{Z}_p$, et que le chiffrement soit *additivement* homomorphe (c'est-à-dire : étant donné c, c' des chiffrés de $m, m' \in \mathbb{Z}_p$, on peut calculer un chiffré du message $m + m'$). Peut-on encore déchiffrer ?
9. Supposons qu'on n'a plus besoin de déchiffrer, mais on veut toujours avoir les propriétés cachant, contraignant, et l'homomorphisme additive. Simplifier ElGamal de manière à ce que $c \in \mathbb{G}$. A-t-on encore besoin d'une hypothèse calculatoire pour les propriétés cachant et contraignant ?
10. RSA est-il cachant et contraignant sans hypothèse calculatoire (resp. avec une hypothèse appropriée) ?

Exercice 3 (Algorithmes génériques de logarithme discret)**.** Considérons un groupe multiplicatif cyclique $\mathbb{G}$ engendré par $g \in \mathbb{G}$ d'ordre premier connu p. Soit h un élément de $\mathbb{G}$. Notre but est de trouver le logarithme discret de h en base g, i.e. trouver $x \in \mathbb{Z}_p$ tel que $g^x = h$.

A. Algorithme naïf. Proposer un algorithme qui trouve x en temps $O(p)$. (Vous avez 30 secondes.)

B. Algorithme de Shanks. Soit $m = \lceil \sqrt{p} \rceil$. Soit x le logarithme discret de h en base g. Soit $x = mq + r$ la division euclidienne de x par m (dans les entiers). On remarque que $hg^{-mq} = g^r$. En se basant sur cette observation, proposer un algorithme qui calcule le logarithme discret dans $\mathbb{G}$ en temps et en mémoire $O(\sqrt{p})$.

C. Algorithme ρ de Pollard. Soit $F : \mathbb{G} \to \mathbb{Z}_p$. Nous définissons une fonction $H : \mathbb{G} \to \mathbb{G}$ par $H(\alpha) = \alpha \cdot h \cdot g^{F(\alpha)}$ et l'algorithme (1). Considérons la suite $(\gamma_i)_{i \geq 1}$ définie par récurrence par $\gamma_1 = h$ et $\gamma_{i+1} = H(\gamma_i)$ pour $i \geq 1$.

1. Montrer qu'à l'entrée de la i-ième boucle **tant que** des lignes 4 à 9 de l'algorithme (1), nous avons
$$\alpha = \alpha_i = g^{x_i} h^i,\ \beta = \beta_i = g^{y_i} h^{2i} \text{ et } x_{2i} = y_i.$$
2. Montrer que si cette boucle termine avec $i < p$ alors l'algorithme retourne le logarithme discret de h en base g.

Algorithme 1 Algorithme ρ de Pollard (pour le logarithme discret)

Entrée: $g, h \in \mathbb{G}$
Sortie: $x \in \{0, \ldots, p-1\}$ tel que $h = g^x$ ou ÉCHEC

$i \leftarrow 1$
$x \leftarrow 0$; $\alpha \leftarrow h$
$y \leftarrow F(\alpha)$; $\beta \leftarrow H(\alpha)$
tant que $\alpha \neq \beta$ **faire**
  $x \leftarrow x + F(\alpha) \bmod p$; $\alpha \leftarrow H(\alpha)$
  $y \leftarrow y + F(\beta) \bmod p$; $\beta \leftarrow H(\beta)$
  $y \leftarrow y + F(\beta) \bmod p$; $\beta \leftarrow H(\beta)$
  $i \leftarrow i + 1$
fin tant que
si $i < p$ **alors**
  renvoyer $(x - y)/i \bmod p$
sinon
  renvoyer ÉCHEC
fin si

3. Soit j le plus petit entier tel que $\gamma_j = \gamma_k$ pour un entier $k < j$. Montrer que $j \leq p + 1$ et que la boucle termine avec $i < j$.
4. Montrer que si F est une fonction aléatoire, alors le temps moyen d'exécution de l'algorithme est en $O(p^{1/2})$ multiplications dans $\mathbb{G}$.

Exercice 4 (Auto-réductibilité du problème du logarithme discret)**.** Soit $\mathbb{G}$ un groupe fini cyclique d'ordre p et g un générateur de $\mathbb{G}$. Considérons un algorithme $\mathcal{A}$ qui prend en entrée un élément de $\mathbb{G}$ et retourne un entier, en temps τ (dans le pire des cas) où τ représente au moins le coût d'une exponentiation dans $\mathbb{G}$.
Supposons qu'il existe un sous-ensemble E de $\mathbb{G}$ avec $|E| \geq \epsilon|\mathbb{G}|$ et $\epsilon \in]0, 1]$ pour lequel lorsque $\mathcal{A}$ est exécuté sur un élément $h \in E$, l'entier retourné par $\mathcal{A}$ est le logarithme discret de h en base g. Considérons l'algorithme $\mathcal{B}$ défini à partir de $\mathcal{A}$ dans l'algorithme (2).

Algorithme 2 Algorithme $\mathcal{B}$

Entrée: $g, h \in \mathbb{G}$
Sortie: $x \in \mathbb{Z}_p$ tel que $h = g^x$.
tant que VRAI **faire**
  $c \xleftarrow{U} \mathbb{Z}_p$ (c est tiré uniformément aléatoirement dans $\mathbb{Z}_p$)
  $h' \leftarrow g^c$
  $w \leftarrow \mathcal{A}(h \cdot h')$
  si $g^w = h \cdot h'$ **alors**
    renvoyer $w - c \bmod p$
  fin si
fin tant que

Montrer que l'algorithme $\mathcal{B}$ résout le problème du logarithme discret dans $\mathbb{G}$ en temps espéré $O(\tau/\epsilon)$.