École normale supérieure **2019-2020**
Département d'informatique

Introduction à la cryptologie
TD n° 3 : RSA, Diffie-Hellman et logarithme discret

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 (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 $\gamma_{i+1} = H(\gamma_i)$

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

1: $i \leftarrow 1$
2: $x \leftarrow 0$; $\alpha \leftarrow h$
3: $y \leftarrow F(\alpha)$; $\beta \leftarrow H(\alpha)$
4: **tant que** $\alpha \neq \beta$ **faire**
5: $\quad x \leftarrow x + F(\alpha) \bmod p$; $\alpha \leftarrow H(\alpha)$
6: $\quad y \leftarrow y + F(\beta) \bmod p$; $\beta \leftarrow H(\beta)$
7: $\quad y \leftarrow y + F(\beta) \bmod p$; $\beta \leftarrow H(\beta)$
8: $\quad i \leftarrow i + 1$
9: **fin tant que**
10: **si** $i < p$ **alors**
11: $\quad$ **retourner** $(x - y)/i \bmod p$
12: **sinon**
13: $\quad$ **retourner** ÉCHEC
14: **fin si**

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
$$\alpha = \gamma_i = g^x h^i \text{ et } \beta = \gamma_{2i} = g^y h^{2i}.$$
2. Montrer que si cette boucle termine avec $i < p$ alors l'algorithme retourne le logarithme discret de h en base g.
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 3. Supposons que l'on a accès à m modules publiques N_i de signatures RSA sur n bits (où m est un grand nombre). À cause de générateurs d'aléa défaillant, on suppose que certaines de ces clefs publiques ont un facteur non-trivial commun. Pour simplifier, on suppose cependant que chaque clef partage au plus un de ses deux exposants avec d'autres clefs (et jamais les deux). On admet que la multiplication, division et PGCD entre deux entiers sur n bits coûtent $\tilde{O}(n)$ opérations de base (où la notation $\tilde{O}$ cache les facteurs polylogarithmiques, i.e. une fonction est $\tilde{O}(f)$ ssi elle est $O(f \cdot \log^k(f))$ pour un certain $k \in \mathbb{N}$).

1. Donner un algorithme qui, étant données deux clefs RSA distinctes avec un facteur commun, calcule la clef de signature de ces deux instances RSA en temps polynomial.
2. Donner un algorithme qui calcule $P = \prod N_i$ en temps $\tilde{O}(mn)$.
3. Donner un algorithme qui calcule $P \bmod N_i^2$ pour tout i en temps $\tilde{O}(mn)$.
4. En déduire un algorithme qui trouve tous les facteurs communs entre les m modules publiques en temps $\tilde{O}(mn)$. (On pourra remarquer que $\gcd(N_i, P/N_i) = \gcd(N_i, (P \bmod N_i^2)/N_i)$.

Exercice 4 (Logarithme discret de petit poids de Hamming)**.** Le nombre de multiplications effectuées par l'algorithme d'exponentiation dichotomique dépend du nombre de 1 dans le développement en base 2 de l'exposant considéré. Il a donc été suggéré pour rendre les protocoles cryptographiques plus efficaces d'utiliser des clés secrètes où ce nombre, le *poids de Hamming* de l'exposant, est relativement petit. Pour se prémunir d'attaque par recherche exhaustive de la clé secrète, il est nécessaire que le nombre de tels exposants soit suffisamment grand. Le nombre d'exposants de ℓ bits de poids w est donné par le coefficient binomial $\binom{\ell}{w}$. L'exercice montre une adaptation de l'algorithme de Shanks pour résoudre le problème du logarithme discret de petit poids de Hamming en $O(\ell\binom{\lceil \ell/2 \rceil}{\lceil w/2 \rceil})$ exponentiations dans le groupe considéré.
Considérons un groupe multiplicatif cyclique $\mathbb{G}$ engendré par $g \in \mathbb{G}$ d'ordre connu q un nombre premier de ℓ bits (*i.e.* $2^{\ell-1} < q < 2^\ell$). Soit w un entier dans $\{1, \ldots, \ell\}$. Nous supposerons que ℓ et w sont pairs.

1. Donner un algorithme pour calculer le logarithme discret dans $\mathbb{G}$ d'un élément h dont le poids de Hamming du logarithme discret est égal à w en $O(\binom{\ell}{w/2})$ exponentiations dans le groupe et dont la complexité en mémoire est $O(\binom{\ell}{w/2})$ éléments de groupe.

Soient N un entier et ℓ et w deux entiers pairs. Un *système de décomposition* de type (N, ℓ, w) est un couple $(X, \mathcal{B})$ tel que
— X est un ensemble de cardinal ℓ ;
— $\mathcal{C}$ est une famille de N sous-ensembles de X de cardinaux $\ell/2$;
— pour tout sous ensemble A de X de cardinal w, il existe un sous-ensemble $C \in \mathcal{C}$ tel que $|A \cap C| = w/2$.

2. Montrer que s'il existe un système de décomposition de type (N, ℓ, w) alors le problème du logarithme discret de poids de Hamming égal à w peut être résolu en $O(N\binom{\ell/2}{w/2})$ exponentiations dans le groupe et en stockant $O(\binom{\ell/2}{w/2})$ éléments de groupe.
3. Posons $X = \mathbb{Z}_\ell$. Montrer que la famille $\mathcal{C} = \{C_i, 0 \leq i \leq \ell/2 - 1\}$, définie par
$$C_i = \{i + j \bmod \ell, 0 \leq j \leq \ell/2 - 1\} \quad \text{pour } 0 \leq i \leq \ell/2 - 1$$
est un système de décomposition de type $(\ell/2, \ell, w)$.
4. Conclure.

Exercice 5 (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**
    $c \xleftarrow{R} \mathbb{Z}_p$ (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**
      retourner $w - c \bmod p$
    fin si
  fin tant que

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