École normale supérieure **2011-2012**
Département d'informatique

Introduction à la cryptologie
TD n° 4 : Cryptographie asymétrique 1/3

Exercice 1. Considérons un groupe multiplicatif cyclique $\mathbb{G}$ engendré par $g \in \mathbb{G}$ d'ordre connu q. Proposer un algorithme de résolution de logarithme discret par compromis temps-mémoire de complexité $O(\sqrt{q})$ opérations de groupe en temps et $O(\sqrt{q})$ éléments de groupe en mémoire.

Indication. On pourra remarquer que pour tout élément $h = g^x \in \mathbb{G}$, l'entier x s'écrit sous la forme $x = x_1 T + x_0$ avec $0 \leq x_1 < T$ et $0 \leq x_2 < T$ pour $T = \lceil \sqrt{q} \rceil + 1$.

Exercice 2. Considérons un groupe multiplicatif cyclique $\mathbb{G}$ engendré par $g \in \mathbb{G}$ d'ordre premier connu q et soit h un élément de $\mathbb{G}$. Soit $F : \mathbb{G} \to \mathbb{Z}_q$. Nous définissons une fonction $H : \mathbb{G} \to \mathbb{G}$ par $H(\alpha) = \alpha \cdot h \cdot g^{F(\alpha)}$ et l'algorithme (1) (appelé *algorithme ρ de Pollard*).

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

Entrée: $g, h \in \mathbb{G}$
Sortie: $x \in \{0, \ldots, q-1\}$ 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: $\quad x \leftarrow x + F(\alpha) \bmod q$; $\alpha \leftarrow H(\alpha)$
6: $\quad y \leftarrow y + F(\beta) \bmod q$; $\beta \leftarrow H(\beta)$
7: $\quad y \leftarrow y + F(\beta) \bmod q$; $\beta \leftarrow H(\beta)$
8: $\quad i \leftarrow i + 1$
9: **fin tant que**
10: **si** $i < q$ **alors**
11: $\quad$ **retourner** $(x - y)/i \bmod q$
12: **sinon**
13: $\quad$ **retourner** ÉCHEC
14: **fin si**

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 que dans la 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 $i < q$ 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 $j \leq q + 1$ 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 $O(q^{1/2})$ multiplications dans $\mathbb{G}$.

Exercice 3. Soit $\mathbb{G}$ un groupe cyclique d'ordre fini n dont la décomposition en facteurs premiers est connue.

1. Montrer que si $n = pq$ est le produit de nombres premiers distincts alors il existe un algorithme qui résout le problème du logarithme discret dans $\mathbb{G}$ en $O(\sqrt{p} + \sqrt{q} + \log(n))$ multiplications dans $\mathbb{G}$.
2. Montrer que si $n = p^e$ où p est un nombre premier et $e \geq 2$ est un entier alors il existe un algorithme qui résout le problème du logarithme discret dans $\mathbb{G}$ en $O(e(\sqrt{p} + \log(n)))$ multiplications dans $\mathbb{G}$.
3. En déduire que si $n = q_1^{e_1} \cdots q_k^{e_k}$ où les q_i sont des nombres premiers deux à deux distincts), alors il existe un algorithme qui résout le problème du logarithme discret en $O\left(\sum_{i=1}^{k} e_i(\sqrt{p_i} + \log(n))\right)$ opérations dans le groupe $\mathbb{G}$.

Exercice 4. Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p engendré par $g \in \mathbb{G}$. De récents cryptosystèmes reposent sur la difficulté des problèmes algorithmiques suivants (pour $\ell \in \mathbb{N} \setminus \{0\}$) :

– Problème ℓ-logarithme discret :
Étant donnés g^x et g^{x^ℓ} dans $\mathbb{G}$ pour x dans $\{1, \ldots, p-1\}$, calculer x

– Problème ℓ-Diffie-Hellman :
Étant donnés g^x et g^{x^ℓ} dans $\mathbb{G}$ pour x dans $\{1, \ldots, p-1\}$, calculer $g^{x^{\ell+1}}$

1. Le problème du 1-logarithme discret est le problème du logarithme discret « classique ». Montrer qu'il existe un algorithme polynomial $\mathcal{A}$ qui résoud le problème Diffie-Hellman « classique » dans $\mathbb{G}$ si et seulement si il existe un algorithme polynomial $\mathcal{B}$ qui résoud le problème 1-Diffie-Hellman dans $\mathbb{G}$.
2. Soit d est un diviseur de $p-1$. En supposant que l'on dispose d'une racine primitive $\zeta \in (\mathbb{Z}/p\mathbb{Z})^*$, donner un algorithme de type « pas-de-bébé, pas-de-géant » qui résoud le problème du d-logarithme discret en $O(\sqrt{d} + \sqrt{p/d})$ exponentiations dans $\mathbb{G}$.

Un nombre composé n est dit pseudo-premier de Fermat en base a si $a^{n-1} \equiv 1 \bmod n$.

Exercice 5. Soit $a \geq 2$ un entier. En considérant les entiers de la forme $(a^{2p}-1)/(a^2-1)$ où p est un nombre premier ne divisant pas a^2-1, montrer qu'il existe une infinité de nombres pseudo-premiers de Fermat en base a.

Un nombre composé n est dit de Carmichael si il est pseudo-premier de Fermat en base a pour tout entier $a > 0$ premier avec n.

Exercice 6.

1. Montrer qu'un nombre de Carmichael est nécessairement impair.

Soient n un nombre de Carmichael et p un facteur premier de n.

2. Montrer que p^2 ne divise pas n.
3. Montrer que $p-1$ divise $n-1$. On pourra considérer une racine primitive a modulo p et montrer que $a^{n-1} = 1 \bmod p$.
4. Réciproquement, montrer que si n est un entier composé sans facteur carré, et tel que pour tout entier p divisant n, $p-1$ divise $n-1$ alors n est un nombre de Carmichael.
5. En déduire qu'un nombre de Carmichael a au moins trois diviseurs premiers.

Exercice 7.

1. Démontrer le critère d'Euler
2. Montrer que si p est premier alors $\mathbb{Z}_{p^t}^*$ est cyclique pour tout entier $t \geq 1$.
3. Montrer que si $n \geq 3$ n'est pas premier alors pour (au moins) la moitié des $a \in \mathbb{Z}_n^*$, nous avons
$$\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \bmod n.$$
4. En déduire pour tout entier T un algorithme qui, prenant en entrée un entier n, retourne Composé ou Premier en $O(T \log^3 n)$ opérations binaires, de sorte que
   – si l'algorithme retourne Composé, alors n est toujours un nombre composé ;
   – si l'algorithme retourne Premier, alors la probabilité que n soit composé est inférieure à 2^{-T}.

Exercice 8. Soient N un module RSA, $2 < e < N$ un entier premier avec $\varphi(N)$ et $2 < d < \varphi(N)$ l'inverse de e modulo $\varphi(N)$. Montrer qu'il existe un algorithme polynomial probabiliste qui, étant donnés N, e et d, retourne la factorisation de N.

Exercice 9.

1. Montrer que si l'on dispose des chiffrés RSA c et c' d'un clair aléatoire m et d'un clair lié $m+r$, où $0 < r < N$ est connu, pour une clé publique $(N, e=3)$, alors on peut retrouver m en temps polynomial.
2. Montrer comment généraliser cette approche pour tout entier e.
3. **Application.** Utiliser la méthode de la question précédente pour retrouver le message m vérifiant $c = m^{17} \bmod N$ et $c' = (m+1)^{17} \bmod N$ avec

$$\begin{aligned} N &= 47502685232865341825439992464725145700424182999231011547935 93 \\ c &= 19356218805125223063783713929395487370916847718680080264316 26 \\ c' &= 10114248818546991018461882489677552339876583926018473780350 75. \end{aligned}$$