Corrigé TD numéro 3

Introduction à la Cryptographie

Préliminaire:

(1) Prouver Solovay-Strassen

Corrigé: Soit n un entier composé de la forme $n = p_1^{k_1} \dots p_r^{k_r}$. On montre que l'ensemble

$$J_n = \{x : (x|p) = x^{(p-1)/2} \bmod p\}$$

dans le cas où n n'est pas premier, J_n est un sous-groupe propre (strict) du groupe multiplicatif $\mathbb{Z}_n^\star$. Il est clair que c'est un sous-groupe à cause des propriétés du symbole de Jacobi: $((ab|n) = (a|n) \cdot (b|n)$ et $(1|n) = 1$, donc $(a^{-1}|n) = (a|n).)$

Pour montrer que c'est un sous-groupe strict, il faut montrer qu'il existe g qui appartienne à $\mathbb{Z}_n^\star$ mais pas à J_n. Soit g l'élément qui vaut g_1 modulo $p_1^{k_1}$, où g_1 est un générateur du groupe cyclique $\mathbb{Z}^\star_{p_1^{k_1}}$, et $g = 1 \bmod m$ où $m = p_2^{k_2} \dots p_r^{k_r}$ et $\gcd(p_1, m) = 1$. Un tel élément existe à cause du théorème des restes chinois.

On a:

$$(g|n) = (g|p_1^{k_1}) \cdot (g|m) = (g_1|p_1^{k_1}).$$

On a $(g_1|p_1^{k_1}) = -1$, sinon g_1 ne serait pas un générateur de $\mathbb{Z}^\star_{p_1^{k_1}}$. Donc, $(g|n) = -1$.

Supposons que $g \in J_n$, alors $(g|n) = g^{(n-1)/2} = -1 \bmod n$. On distingue deux cas.

Premier cas: $k_1 > 1$. Alors, $g_1^{n-1} = 1 \bmod p_1^{k_1}$ et donc, $p_1^{k_1-1}(p_1 - 1)|(n-1)$ car l'ordre de $\mathbb{Z}^\star_{p_1^{k_1}}$ est $\varphi(p_1^{k_1}) = p_1^{k_1-1}(p_1 - 1)$. De plus, $p_1|n$ et donc, ce n'est pas possible qu'un entier p_1 divise n et que p_1 divise $n - 1$. Contradiction et donc $g \notin J_n$.

Deuxième cas: $k_1 = 1$. Si $J_n = \mathbb{Z}_n^\star$, alors $g^{n-1} = -1 \bmod n$ et comme $m|n$, alors $g^{n-1} = -1 \bmod m$. On a immédiatement une contradiction car $g = 1 \bmod m$.

(2) Montrer que pour tout entier $m \in \mathbb{Z}_N$, $m^{ed} = m \bmod N$.

Corrigé: D'après le théorème d'Euler, on a pour tout $m \in \mathbb{Z}_N^\star$, $m^{\varphi(N)} = 1 \bmod N$ et donc

$$m^{ed} = m^{1+k\varphi(N)} = m \cdot (m^{\varphi(N)})^k = 1 \bmod N$$

Il reste donc à prouver le résultat pour $m \notin \mathbb{Z}_N^\star$, autrement dit pour m un multiple de p et ou q. Pour $m = \lambda \times p$, (de manière similaire pour $m = \mu \times q$), on a $m = 0 \bmod p$ et $m = m_q \bmod q$. Alors, $m^{ed} = 0 \bmod p$ et

$$m^{ed} = (m_q)^{1+k\cdot\varphi(N)} = m_q^{1+k'(q-1)} = m_q \bmod q$$

car d'après le petit théorème de Fermat, $m_q^{q-1} = 1 \bmod q$ pour $m_q \neq 0 \bmod q$. (Le cas $m_q = 0$ est facile à traiter.) On voit alors que $m^{ed} = 0 = m \bmod p$ et $m^{ed} = m_q = m \bmod q$ et donc d'après le théorème des restes chinois, il existe un unique entier modulo pq qui vérifie ces deux équations qui est m.

Exercice 1: Factorisation - Méthode $P+1$

Supposons que p soit un facteur de N et qu'il soit friable (n'ait que des petits facteurs $\leq B$).

Considérer le corps fini à p^2 éléments de caractéristique p. Soit P un entier et supposons que P^2-4 soit un non-résidu quadratique modulo p. Montrer que si α est l'une des deux racines du polynôme $f(X)=X^2-PX+1$, (l'autre α^{-1}), alors α^p est aussi racine de ce polynôme.

Montrer alors que $\alpha^{p+1}=1$ et en conclure un algorithme pour factoriser les nombres N de cette forme.

Corrigé:

Comme $P^p = P \bmod p$, alors α^p est une racine de f. En effet,

$$f(\alpha^p) = \alpha^{2p} - P\alpha^p + 1 = (\alpha^2 - P\alpha + 1)^p = (f(\alpha))^p = O \text{ dans } GF(p^2)$$

Comme f a seulement 2 racines: $\alpha^p = \alpha$ ou $\alpha^p = \alpha^{-1}$. Si $\alpha^p = \alpha$, alors $\alpha \in GF(p)$ et $P^2-4 = (\alpha+\alpha^{-1})^2 - 4 = (\alpha-\alpha^{-1})^2$, ce qui n'est pas possible car P^2-4 n'est pas un résidu quadratique et $\alpha - \alpha^{-1}$ est un élément de $GF(p)$ (car si $\alpha \in GF(p)$, alors α^{-1} aussi). Par conséquent, $\alpha^p = \alpha^{-1}$ et

$$\alpha^{p+1} = 1.$$

La méthode $P-1$ choisit a dans le groupe multiplicatif de $\mathbb{Z}_p^\star$ d'ordre $p-1$ et retrouve p si son ordre est lisse (smooth). La méthode $P+1$ est similaire à la méthode $P-1$ et réussit si l'ordre de α est lisse.

En effet, si $p+1$ est lisse, alors l'ordre de α, e est lisse lui aussi. Si on calcule α^e, on doit obtenir 0 dans $GF(p^2)$ et $\neq 0$ dans $GF(q^2)$. Donc les deux composantes de $\alpha^e - 1$ en tant que polynôme linéaire à coefficients dans $GF(p)$ sont nulles et non nulles dans $GF(q)$. En calculant $\gcd(N, (\alpha^e-1)_0)$, on obtient p où $a(x)_0$ est le coefficient constant du polynôme $a(x)$.

On effectue les calculs modulo N car p n'est pas connu. Les calculs peuvent être effectués de manière plus efficace si on manipule les fonctions de Lucas $\alpha^e + \alpha^{-e}$ à la place de α^e.

Exercice 2: RSA et Factorisation

(1) Montrer si $m' = m + r$ avec r connus et $c = m^3 \bmod N$ et $c' = m'^3 \bmod N$, alors on peut calculer m ?

(2) Montrer que si on connaît d_p et d_q, on peut factoriser N.

(3) Montrer que si on connaît d_p, alors on peut factoriser N ?

(4) Montrer comment déchiffrer si le message $m < 2^{64}$ par exemple avec e grand.

Corrigé:

(1) On vérifie que $c' = m'^3 = (m+r)^3 = m^3 + 3mr^2 + 3mr^2 + r^3 \bmod N$ et donc

$$c' - c + 2r^3 = r \cdot (3m^2 + 3mr + 3r^2) \bmod N.$$

De plus,

$$c' + c - r^3 = m \cdot (3m^2 + 3mr + 3r^2) \bmod N$$

et donc

$$\frac{c' - c + 2r^3}{c' + c - r^3} = \frac{r}{m}.$$

(2) On a

$$d_p = 1 \bmod (p-1) \text{ et } d_q = 1 \bmod (q-1)$$

et donc, il existe deux entiers k et ℓ tels que

$$d_p = 1 + k(p-1) \text{ et } d_q = 1 + \ell(q-1).$$

Par conséquent, on obtient un multiple de $\varphi(N) = (p-1)(q-1)$ en calculant $(d_p - 1) \cdot (d_q - 1)$. On a vu en cours qu'à partir d'un multiple de $\varphi(N)$ on savait factoriser en utilisant la même astuce que dans le test de primalité de Miller-Rabin. On trouve sur une racine non triviale de l'unité mod$N$ en calculant les carrés successifs de a^t où t est la partie impair du multiple de $\varphi(N)$ et a un élément de $\mathbb{Z}_N^\star$, avec probabilité $1/2$. Avec cette racine r, en calculant $\gcd(r-1, N)$ on arrive à factoriser N.

(3) Si on n'a que d_p, il suffit de remarquer qu'on a en fait un multiple de $p-1$ et que la méthode $P-1$ utilise la même idée. En calculant a^{p-1}, cette valeur vaut 1 mod p et $\neq 1$ mod q et donc $\gcd(a^{p-1} - 1, N) = p$.

Exercice 3: Fonction de Rabin

(a) Montrer que si p est un nombre premier impair et a un carré modulo p, alors il existe un algorithme A probabiliste en temps polynomial en moyenne tel que $A(p, a) = x$ tel que $x^2 = a \bmod p$.

(b) Soit p et q sont premiers, $n = pq$ et a un carré modulo p. Donner un algorithme probabiliste et en temps polynomial en moyenne tel que $A'(p, q, n, a) = x$ où $x^2 = a \bmod n$.

(c) Montrer que calculer des racines carrés modulo n est aussi difficile que factoriser.

(d) En déduire une fonction à sens unique à trappe.

(e) Montrer que la self-random réducibilité de ce problème.

(f) Montrer que la fonction $f : x \mapsto x^2 \bmod n$ de QR_n vers QR_n où QR_n est l'ensemble des résidus quadratiques modulo n et $n = pq$ où $p = q = 3 \bmod 4$ est une permutation. On a donc une permutation à sens unique à trappe. Son inverse est la fonction $g(x) = x^{((p-1)(q-1)+4)/8} \bmod n$.

Corrigé:

(a) Si $p = 3 \bmod 4$, alors on sait qu'en calculant $a^{\frac{p+1}{4}}$, on trouve une racine carrée de a car

$$\left(a^{\frac{p+1}{4}}\right)^2 = a^{\frac{p-1}{2}+1} = a \cdot (a|p) = a$$

car $(a|p) = 1$ puisque a est un carré.

Si $p = 1 \bmod 4$, il existe aussi un algorithme mais qui est un peu plus compliqué. Un sous-cas est $p = 5 \bmod 8$ et le dernier sous-cas $p = 1 \bmod 8$ est laissé au lecteur (il s'agit d'une extension de ce dernier cas). Si $p = 5 \bmod 8$, alors $(p+1)/2 = 4k + 3$ est impair et l'idée précédente ne s'applique plus. Cependant, on sait que $a^{4k+2} = 1 \bmod p$ et a^{2k+1} est une racine de l'unité. Si $a^{2k+1} = 1$, alors on a terminé par le même argument que précédemment. Le problème est qu'il se peut aussi que $a^{2k+1} = -1 \bmod p$. C'est

ici qu'il faut utiliser le fait qu'on peut trouver avec un algorithme probabiliste un non-résidu quadratique b. Comme $(p-1)/2 = 4k+2$, le symbole de Legendre de b est $b^{4k+2} = -1$, ce qui implique que $a^{2k+1}b^{4k+2} = 1 \bmod p$ ou de manière équivalente,

$$a^{2k+2}b^{4k+2} = a \bmod p.$$

Comme les deux exposants de gauche sont paires, on peut conclure que $\pm a^{k+1}b^{2k+1} \bmod p$ sont les 2 racines carrées de a.

Pour le dernier cas, $p = 1 \bmod 8$, il faut écrire $p = 8k+1$ et $k = 2^s t$ où t est impair. Si $a^t = 1 \bmod p$, alors les racines sont $\pm a^{\frac{t+1}{2}} \bmod p$ et sinon il faut trouver le plus petit indice i où $a^{2^i t} \neq 1 \bmod p$ et terminé en utilisant un non-résidu quadratique.

(b) Si on connaît les facteurs p et q de n, il suffit de calculer une des 2 racines de a modulo p et modulo q et de recombiner avec les restes chinois. Tous ces algorithmes sont en temps polynomial et probabilistes.

(c) Si on suppose qu'on a un algorithme qui étant donné n, retrouve p et q, alors on peut utiliser l'algorithme A' pour trouver les racines carrées. Réciproquement, si on a un algorithme A qui retrouve les racines carrées, il suffit de choisit a aléatoirement, calculer $a^2 \bmod n$ et avec probabilité $1/2$, l'algorithme va nous retourner une racine carrées qui sera par exemple $a_1 = -a \bmod p$ et $a_1 = a \bmod q$ ou $a_2 = a \bmod p$ et $a_2 = -a \bmod q$. L'une de ces deux racines nous permet de factoriser car $\gcd(n, a + a_1) = p$.

(d) La fonction $f(x) = x^2 \bmod N$ est une fonction à sens unique à trappe sous l'hypothèse que la factorisation est un problème difficile où la trappe est la factorisation de N.

(e) Supposons que l'on sache inverser la fonction f sur une fraction ϵ de $\mathbb{Z}_n^\star$, montrons comment inverser la fonction pour tout $x \in \mathbb{Z}_n^\star$. L'idée est d'utiliser le fait que si x n'est pas dans l'ensemble qu'on sait inverser, alors $z = xy^2 \bmod n$ le sera. Si on recommence suffisamment souvent, on va tomber sur un z pour lequel on sera calculer une racine carrée z_1 et une racine carrée de x sera z_1/y.

On doit aussi montrer que si y est choisi uniformément dans $\mathbb{Z}_n^\star$, alors $z = xy^2 \bmod n$ parcourt aussi le sous-groupe des carrées de manière uniforme et donc chaque fois qu'on calcule un z, on a une probabilité ϵ de tomber sur un élément pour lequel on sait calculer une racine carrée. Ainsi, après $1/\epsilon$ essais, on va savoir calculer une racine carrée de x.

(f) Pour montrer que c'est une permutation, montrons que la fonction est injective. Si $f(x_1) = f(x_2)$, alors $x_1^2 = x_2^2 \bmod n$ et donc $x_1 = \pm x_2 \bmod p$ et $x_1 = \pm x_2 \bmod q$. On sait que -1 n'est pas un carré modulo p et modulo q car $(-1|p) = (-1)^{(p-1)/2} = -1$ car $(p-1)/2$ et $(q-1)/2$ sont impairs. Par conséquent, il n'y a qu'une seule des 4 racines carrées qui sera aussi un carré.

Exercice 4: Logarithme Discret

Le but de l'exercice est de résoudre le problème du logarithme discret dans deux cas particuliers:

1. quand le logarithme s'écrit comme le produit de deux entiers
2. quand le poids de Hamming du logarithme est petit.

Soit G un groupe cyclique abélien d'ordre n généré par un élément g et noté multiplicativement. Par conséquent, $G = \{g^i : i \in [0, n-1]\}$. Pour toute valeur v de G,

le *logarithme discret* de v en base g, noté $\log_g(v)$ est l'unique entier x compris entre 0 et $n-1$ tel que $v = g^x$. Sauf mention contraire, on suppose que l'ordre n est connu. Le poids de Hamming d'un entier x, codé sur ℓ bits, est le nombre de 1 dans la représentation binaire de x.

1) Expliquer comment fonctionne l'algorithme Baby-Step Giant-Step et donner sa complexité en temps en nombre de multiplications modulaires et en nombre d'éléments de G stockés en mémoire. On rappelle que cet algorithme cherche deux entiers i et j tels que $\log_g(v) = i + jm$ où $m = \lceil n^{1/2} \rceil$.

Corrigé: C'est du cours. On fait $2\sqrt{n}$ multiplications modulaires et le nombre d'éléments stockés en mémoire est $\sqrt{n}$.

2) Proposer une variante de l'algorithme précédent quand on sait que le logarithme discret x s'écrit comme le produit de deux entiers $x_1 \in X_1$ et $x_2 \in X_2$. Donner sa complexité en temps et mémoire en fonction de la taille de X_1 et X_2, notée $|X_1|$ et $|X_2|$.

Corrigé: On écrit $v = g^{x_1 x_2}$ et on calcule pour tout $x_1 \in X_1$, $v \times g^{1/x_1}$ qu'on stocke dans une table et on calcule à la volée pour tout $x_2 \in X_2$, g^{x_2}. Pour chacun de ces éléments, on vérifie s'il est dans la table. On aura bien sûr pris soin de trier la table des $v \times g^{1/x_1}$ par valeur croissante pour que la recherche s'effectue en temps $O(\log(X_1))$. Il faut stocker l'une des deux valeurs en fonction de la taille de X_1 et X_2.

3) On suppose toujours que le logarithme s'écrit comme le produit de deux entiers $x = x_1 \times x_2 \bmod n$ où $x_i \in X_i$ pour $i = 1, 2$. Quel problème se pose si l'ordre n n'est pas connu ?

Corrigé: Si l'ordre n n'est pas connu, alors on ne sait pas inverser $x_1 \bmod n$ et on ne sait pas calculer $v \times g^{1/x_1 \bmod n}$!

(Difficile) Montrer que si $v = g^x$, alors $v^\pi = g^{x\pi}$ où $\pi = \prod_{i \in X_1} i$. Donner un algorithme dont la complexité en temps est en $O((|X_2| + |X_1| \log_2 |X_1|) \log_2 n)$ multiplications d'éléments du groupe.

Corrigé: Cependant, pour éviter le problème d'avoir à calculer l'inverse, on peut multiplier par $\prod_{i \in X_1} i$ pour ne pas avoir de division à effectuer. On voit alors qu'en multipliant par cette valeur dans les exposants, on a tout de même une égalité et donc il suffit de diviser cette égalité. Mais si on divise maintenant, on se retrouve avec des entiers dans les exposants et on n'a plus le problème qu'on avait au début quand l'ordre n est inconnu. On construit les ensembles

$$S_1 = \{v^{\prod_{k \in X_1 \setminus \{i\} k}; \forall i \in X_1}\} \text{ et } S_2 = \{g^{\pi \times j}; \forall j \in X_2\}.$$

On peut construire S_1 de manière à construire un arbre, on part de v, puis on calcule 2 noeuds: $v^{x_3 x_4}$ et $v^{x_1 x_2}$. Pour le premier noeud, on construit les 2 noeuds: $v^{x_2 x_3 x_4}$ et $v^{x_1 x_3 x_4}$ et pour le second noeud on construit les deux noeuds: $v^{x_1 x_2 x_4}$ et $v^{x_1 x_2 x_3}$. On voit que si on part d'un ensemble initial de 2^q valeurs, on aura besoin de calculer, 2^q valeurs et qu'on pourra le faire en temps $q \times 2^q$ exponentiations.

Soit deux entiers t et ℓ tels que $0 < t < \ell$. Un (t, ℓ)-*système de découpage* est une paire $(X, \mathcal{B})$ tels que:

- $|X| = \ell$, et $\mathcal{B}$ est un ensemble de sous-ensembles de X, chaque sous-ensemble ayant $\ell/2$ éléments.
- $\forall Y \subset X$, tel que $|Y| = t$, $\exists\ B \in \mathcal{B}$ tel que $|B \cap Y| = t/2$.

4) Montrer que si $X = [0, \ell - 1]$ et $\mathcal{B} = \{B_i : 0 \leq i \leq \ell/2 - 1\}$ où pour tout $0 \leq i \leq \ell/2 - 1$, $B_i = \{i + j : 0 \leq j \leq \ell/2 - 1\}$, alors $(X, \mathcal{B})$ est un (t, ℓ)-système de découpage.

Corrigé: Il est facile de vérifier les deux assertions.

5) Trouver un algorithme pour calculer le logarithme discret d'un élément dont son poids de Hamming est $\leq t$. Sa complexité en temps doit être en $O(\ell\binom{\ell/2}{t/2})$ exponentiation dans le groupe G et $O(\binom{\ell/2}{t/2})$ éléments de G en mémoire.

Corrigé: Soit v un élément du groupe engendré par g dont le logarithme discret est de poids de Hamming t (supposons-le pair). Tout élément de $\langle g \rangle$ peut être représenté par les non zero bits et v s'identifie à un sous-ensemble Y de X de t éléments. Le but de l'algorithme est de trouver une décomposition de Y en 2 sous-ensembles de $t/2$ éléments en utilisant le système de découpage.

Pour toute valeur B_i de $\mathcal{B}$:

- Pour tout $Y_i^j \subset B_i$ de $t/2$ éléments, identifier la valeur A_i^j de G. Soit L_1 la liste des paires (Y_i^j, A_i^j).
- Considérer l'ensemble $W_i = [0, \ell - 1] \setminus B_i$.
- Pour tout $W_i^k \subset W_i$ de $t/2$ éléments, identifier la valeur B_i^k correspondante dans G. Soit L_2 la liste des paires $(W_i^k, v \times (B_i^k)^{-1})$.
- Si deux valeurs $A_i^{j_0}$ de L_1 et $v \times (B_i^{k_0})^{-1}$ de L_2 sont égales pour un ensemble donné $Y_i^{j_0}$ et un ensemble donné $W_i^{k_0}$, alors retourner l'élément de $\langle g \rangle$ identifié par $Y_i^{j_0} \cup W_i^{k_0}$; sinon aller à l'iteration suivante.

La complexité en temps est $O(\ell\binom{\ell/2}{t/2})$ exponentiations et en espace $O(\binom{\ell/2}{t/2})$.