Cryptographie

Ibtissem Zaafrani
sous la direction de Michaël Bulois

June 30, 2014

Introduction.

Parfois, il est important de pouvoir transmettre des informations secrètes qui ne puissent pas être compris que par le destinataire. La cryptographie est un outil qui permet de protéger les messages. Elle est utilisée depuis l'Antiquité.

Dans la section 1, on présentera le principe de la cryptographie en se basant sur un exemple d'un système cryptographique classique nommé chiffrement de Vigenère. Dans la section 2, on introduira le problème du logarithme discret et on étudiera le chiffrement ElGamal qui est basé sur ce problème. On introduira également dans la section 3 la notion de complexité qui sert à déterminer le temps de calcul d'obtention du résultat cherché. La dernière section sera consacrée à l'algorithme de Shanks qui permet de calculer le logarithme discret.

Les références principales pour ce travail ont été [Sti] et [Ste].

1 Principe et exemple

1.1 Introduction à la cryptographie

La cryptographie est l'étude des méthodes permettant de transmettre des données de manière confidentielle sur un canal non sécurisé. Cette transformation se fait en deux étapes : La première, qu'on appelle chiffrement, transforme un texte clair donné en un texte chiffré incompréhensible. Inversement, la deuxième, qu'on appelle déchiffrement, permet de retransformer le texte chiffré en un texte clair. Ces deux transformations utilisent un paramètre appelé clé.

Chaque système cryptographique est caractérisé par son ensemble fini possible de blocs de textes clairs, de blocs de textes chiffrés et de clés et des fonctions de chiffrement et de déchiffrement associées à chaque choix de clé. Si Alice souhaite envoyer le texte clair x à Bob, elle choisit une fonction de chiffrement définie sur l'ensemble des blocs de textes clairs et elle l'applique en chaque bloc de x. Après elle envoie le texte chiffré obtenu à Bob. Lorsque Bob reçoit le texte chiffré, il le déchiffre à l'aide d'une fonction de déchiffrement définie sur

l'ensemble des blocs de textes chiffrés de telle sorte qu'il obtienne le texte x. La fonction de chiffrement doit être injective car, sinon, Bob aura plusieurs textes clairs correspondants au même texte chiffré.

On distingue deux types de systèmes cryptographiques: système à clé secrète (symétrique) et système à clé publique (asymétrique). Si les clés de chiffrement et de déchiffrement sont identiques, le système alors est dit à clé secrète. Si elles sont distinctes et qu'on ne peut pas déduire l'une de l'autre alors le système est dit à clé publique.

1.2 Vigenère

Le chiffrement de Vigenère est un système cryptographique à clé secrète, c'est à dire qu'il utilise la même clé pour le chiffrement et le déchiffrement.

Voici comment fonctionne le chiffrement de Vigenère :
Premièrement, on choisit une clé. Cette clé est composée de lettres, on l'appelle *mot-clef*. Il est important qu'elle soit connue seulement par l'expéditeur et le destinataire car, sinon, n'importe quel autre personne peut décrypter le texte chiffré. Ensuite, on convertit chaque lettre du message et du mot-clef en un chiffre de 0 à 25 comme le montre le tableau suivant :

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Après, on regroupe les chiffres du texte clair par blocs de même longueur que la clé. Enfin, on fait un décalage à chaque bloc par le mot-clef, c'est à dire on décale le i-éme nombre de chaque bloc par le i-éme nombre du mot-clef modulo 26.

Pour le déchiffrement, on réalise l'opération inverse de chiffrement. C'est-à-dire, on soustrait la clé au texte chiffré.

Prenons un exemple :
supposons que le texte clair soit VIGENERE et que BON soit le mot-clef.
La clé est converti en (1,14,13).
Le texte est converti en (21,8,6,4,13,4,17,4). On le divise en bloc de longeur 3:

$$(21,8,6)\ \ (4,13,4)\ \ (17,4)$$

Après un décalage modulo 26, on obtient

$$(22,22,19)\ \ (5,1,17)\ \ (18,18)$$

et le texte chiffré est donc

WWTFBRSS .

On remarque que la lettre E a été chiffré en F, R et S . On parle alors d'un système polyalphabétique.
Le nombre de mots-clef possibles dans ce système est 26^m où m la longueur de la clé. Donc, plus la clé est longue, plus le code sera difficile à décrypter par un attaquant.

2 Logarithme discret et chiffrement ElGamal

2.1 Un peu de théorie des groupes

Définition 1. *: Soit $(G,.)$ un groupe fini et $\alpha \in G$. Le sous groupe de G engendré par α est défini par $<\alpha>=\{\alpha^m, m \in \mathbb{N}\}$.*

Le plus petit entier naturel non nul m qui vérifie $\alpha^m = 1$ est appelé ordre de α et on le note $ord(\alpha)$ ou $o(\alpha)$.

Définition 2. *Soit $(G,.)$ un groupe fini d'ordre n. On dit que G est cyclique s'il existe $\alpha \in G$ tel que $G =<\alpha>$.*

Autrement dit, pour tout $\beta \in G$ il existe k compris entre 0 et $n-1$ tel que $\alpha^k = \beta$.

Cet élément α est appelé générateur ou primitif de G.
On rappelle que $(\mathbb{Z}/p\mathbb{Z})^* = \{\overline{1}, ..., \overline{p-1}\}$ est un groupe multiplicatif commutatif.

Théorème 1. *Si p un nombre premier alors le groupe $((\mathbb{Z}/p\mathbb{Z})^*, \times)$ est cyclique.*

Pour la démonstration on va utiliser ce lemme:

Lemme 1. *Soit $(G,.)$ un groupe commutatif fini et soit n le ppcm des ordres des éléments de G. Alors G admet un élément d'ordre n.*

Démonstration. On factorise n en produit de nombres premiers : $n = p_1^{\ell_1}...p_s^{\ell_s}$. Pour tout $i \in \{1, ..., s\}$, il existe $x_i \in G$ tel que $x_i^{n/p_i} \neq 1$ car, sinon, n/p_i est un multiple de tous les ordres des éléments de G et donc de n. Posons $\gamma_i = \prod_{\substack{j=1 \\ j\neq i}}^{s} p_j^{\ell_j}$ avec $\gamma_i = 1$ si $s = 1$ et soit $y_i = x_i^{\gamma_i}$.

On a $y_i^{p_i^{\ell_i}} = (x_i^{\gamma_i})^{p_i^{\ell_i}} = x_i^n = 1$ d'après la définition de n. Ainsi $p_i^{\ell_i}$ est un multiple de l'ordre de y_i. Or

$$y_i^{p_i^{\ell_i - 1}} = (x_i^{\gamma_i})^{p_i^{\ell_i - 1}} = x_i^{n/p_i} \neq 1.$$

On obtient $ord(y_i) = p_i^{\ell_i}$. Soit m l'ordre de $z = y_1...y_s$. Comme G est commutatif, alors $y_1^m...y_s^m = 1$ et donc $y_i^m = \prod_{\substack{j=1 \\ j\neq i}}^{s} y_j^{-m}$.

D'où $y_i^{m\gamma_i} = \prod_{\substack{j=1 \\ j\neq i}}^{s} y_j^{-m\gamma_i} = 1$ car γ_i est un multiple de l'ordre de y_j pour $j \neq i$. L'entier $m\gamma_i$ est donc un multiple de $p_i^{\ell_i}$. Mais $p_i^{\ell_i}$ et γ_i sont premiers entre eux, d'après le théorème de Gauss $p_i^{\ell_i} | m$ et donc $n|m$. Comme $z^n = 1$ alors $n = m$. □

Démonstration. (Théorème 1) :
Soit $G = (\mathbb{Z}/p\mathbb{Z})^*$ et n le ppcm des ordres des éléments de G.

Posons $P(X) = X^n - \overline{1}$ un pôlynome de $(\mathbb{Z}/p\mathbb{Z})[X]$. Pour tout $a \in G$, on a $P(a) = 0$ car $a^n = 1$. Donc P admet au minimum $p-1$ zéros sur le corps $\mathbb{Z}/p\mathbb{Z}$ et comme son degré est n alors $p-1 \leq n$. D'après Lagrange l'ordre de tout élément de G divise $p-1$ et du lemme précédent $n|p-1$. On en déduit que $n = p-1$.

D'ou le résultat. □

2.2 Problème du logarithme discret

Soit p un grand nombre premier et soit α un générateur de $(\mathbb{Z}/p\mathbb{Z})^*$. L'existance de α est assuré par le théorème 1.

Problème. *Etant donné β élément de $(\mathbb{Z}/p\mathbb{Z})^*$, trouver l'unique exposant a, $0 \leq a \leq p-2$, tel que $\alpha^a \equiv \beta \pmod{p}$.*

On dit alors que a est le logarithme discret de β en base α dans $(\mathbb{Z}/p\mathbb{Z})^*$ et on le note $log_\alpha \beta$.
Le fait que p soit un grand nombre est très important pour que le problème du logarithme discret soit difficile.

Remarque. Le problème du logarithme discret peut s'appliquer à n'importe quel groupe cyclique G autre que $(\mathbb{Z}/p\mathbb{Z})^*$.
Plus généralement, tout $\beta \in G$ possède un logarithme discret en base α ssi G est cyclique de générateur α.

Un des algorithmes cryptographiques qui est basé sur ce logarithme et qu'on va étudier dans la suite est le chiffrement ElGamal. On s'intéresse dans ce chiffrement au cas $G = (\mathbb{Z}/p\mathbb{Z})^*$.

2.3 Chiffrement ElGamal

Le chiffrement ElGamal est un système cryptographique à clé publique qui repose sur le problème du logarithme discret. Il a été créé par Taher Elgamal en 1985.
Le principe de ce système est le suivant:
Alice veut envoyer le message m à Bob. Tout d'abord Bob choisit un grand nombre premier p et un générateur α de $(\mathbb{Z}/p\mathbb{Z})^*$. Il choisit aussi et aléatoirement un entier a compris entre 0 et $p-2$. Ensuite Bob calcule $\alpha^a \equiv \beta \pmod{p}$. Alors Bob diffuse la clé publique (p, α, β) et garde sa clé secrète a (on l'appelle clé privée).
Soit n le plus grand entier qui vérifie $26^n \leq p$.
Pour le chiffrement, Alice divise le message m en blocs de longueur n. Ensuite, elle convertit chaque bloc en un entier n_i modulo p. Alice choisit aléatoirement un nombre $k \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$ secret. Après elle calcule, pour tout i,

$$\xi_1 \equiv \alpha^k \pmod{p} \text{ et } \xi_{2,i} \equiv n_i \beta^k \pmod{p}.$$

Enfin elle envoie le texte chiffré (ξ_1, ξ_2) à Bob où $\xi_2 = (\xi_{2,1}, \xi_{2,2}, ...)$.
Pour le déchiffrement, Bob utilise sa clé secrète a et calcule

$$\xi_{2,i}({\xi_1}^a)^{-1} \equiv n_i \pmod{p}.$$

Finalement, Bob convertit chaque n_i en un bloc de lettres et retrouve le message m.

Justification. L'élement $({\xi_1}^a)^{-1}$ existe. En effet, on calcule d'abord ${\xi_1}^a \pmod{p}$ puis on calcule son inverse modulo p . L'inverse existe car on travaille dans $(\mathbb{Z}/p\mathbb{Z})^*$ avec p premier.

$$\begin{aligned}
\xi_{2,i}({\xi_1}^a)^{-1} &\equiv \xi_{2,i}((\alpha^k)^a)^{-1} \pmod{p} \\
&\equiv \xi_{2,i}(\alpha^{ka})^{-1} \pmod{p} \\
&\equiv n_i \beta^k (\beta^k)^{-1} \pmod{p} \\
&\equiv n_i \pmod{p}
\end{aligned}$$

Le chifffrement ElGamal est non déterministe, car l'étape de chiffrement dépend, en plus du choix de m, d'une valeur aléatoire k choisie par Alice. On trouve donc plusieurs textes chiffrés qui correspondent au même texte clair choisi.

Exemple. Soient $p = 541$, $\alpha = 2$, $a = 7$ et $k = 15$. Alors

$$\beta \equiv 2^7 \pmod{541} = 128$$

Supposons qu'Alice veuille envoyer le message $x = 42$ à Bob. Elle calcule

$$\xi_1 \equiv 2^{15} \pmod{541} = 308$$

ensuite

$$\xi_2 \equiv 42 \times 128^{15} \pmod{541} = 252.$$

Lorsque Bob reçoit le texte chiffré, il calcule

$$(308^7)^{-1} \equiv 6^{-1} \pmod{541} \equiv 451 \pmod{541}$$

puis

$$252 \times 451 \pmod{541} = 42.$$

Remarque. On peut généraliser l'algorithme de ElGamal sur un groupe cyclique G de générateur α tel que le problème du logarithme discret soit difficile.

3 Complexité

3.1 Ecriture d'un entier dans une base

Soit $m \in \mathbb{N}^*$, soit $b \geq 2$. m s'écrit en base b avec k chiffres si et seulement si il existe $a_0, a_1, ..., a_{k-2}, a_{k-1} \in \{0, ..., b-1\}$ tels que

$$m = a_{k-1}b^{k-1} + a_{k-2}b^{k-2} + ... + a_1 b + a_0.$$

Cette écriture implique que $b^{k-1} \leq m < b^k$. Donc $k - 1 \leq log_b(m) < k$. Ainsi pour écrire m en base b il nous faut $\lfloor log_b(m) \rfloor + 1$ chiffres. D'ou l'unicité de k. Les entiers a_0,...,a_{k-1} sont obtenu comme restes des divisions successives par b. On peut donc définir l'écriture dans une base comme suivant :

Définition 3. *Soit $m \in \mathbb{N}^*$, soit $b \geq 2$. L'écriture de m en base b est*

$$\overline{a_{k-1}a_{k-2}...a_0}^b.$$

Où les a_i, pour $0 \leq i \leq k-1$, sont déterminés comme précédemment.

3.2 Définition et premier calcul

L'objectif de la cryptographie est d'assuer la securité des messages. Mais, dans notre cas, il est toujours possible pour l'attaquant de tester toutes les possibilités afin de retrouver la clé secrète et ainsi décrypter tous les textes chiffrés. Notre but est de rendre ce décryptage plus difficille pour l'attaquant en terme de temps que pour le déstinataire. Nous allons donc nous intéresser à la complexité de ces opérations.

Définition 4. *La complexité d'un problème donné c'est le nombre d'opérations binaires qui interviennent dans la résolution de ce problème.*

Exemple. Déterminons le nombre d'opérations dans le calcul de $\overline{11}^2 + \overline{10}^2$:

$$\begin{array}{r} 11 \\ +_1 10 \\ \hline 101 \end{array}$$

On calcule d'abord le chiffre des unités. On a donc une opération binaire $1 + 0$. Puis, on calcule $1 + 1$. On a alors effectué une autre opération binaire. Puisque $1 + 1 = 2 = \overline{10}^2$ on aura aussi une autre opération binaire pour la retenue. Finalement, on a trois opérations binaires.

Pour simplifier les calculs, on s'intéresse aux ordres de grandeur grace à la définition suivante.

Définition 5. *Soient f et g deux fonctions de $\mathbb{N}^*$ à valeur dans $\mathbb{R}^+$. On dit que $f(n) = O(g(n))$ s'il existe $M > 0$ et $n_0 \in \mathbb{N}$ tels que pour tout $n \geq n_0$ $f(n) \leq Mg(n)$.*

On travaille dans la suite en base 2 et on note $log = log_2$.

Proposition 1. *Soient a et b deux éléments de $\mathbb{Z}/p\mathbb{Z}$.*
Alors

1. *$a+b \pmod p$ se calcule en $O(log(p))$.*
2. *$a.b \pmod p$ se calcule en $O((log(p))^2)$.*

Preuve.

Soient $a_1, b_1 \in [\![0, p-1]\!] \subset \mathbb{Z}$ les représentants respectifs de $a, b \in \mathbb{Z}/p\mathbb{Z}$. Supposons que $a_1 \geq b_1$ et soit $k = \lfloor log(a_1) \rfloor + 1$.

1. On calcule $c = a_1 + b_1$: on effectue au maximum k opérations binaires et au maximum k retenues. Si $c \leq p$, alors on a $O(2.log(p)) = O(log(p))$ opérations binaires. Sinon on calcule $c-p$ qui est de complexité $O(log(p))$. On aura donc $O(3.log(p)) = O(log(p))$ opérations binaires.

2. Elle se fait en deux étapes :
   • On multiplie a_1 par chaque chiffre de b_1. On a $log(p)O(log(p)) = O((log(p))^2)$ opérations binaires avec au maximum $k.O(log(p)) = O((log(p))^2)$ retenues.
   • Pour la deuxième étape on additionne les lignes deux à deux. On a au maximum k lignes et d'après 1) on a $k.O(log(p)) = O((log(p)^2))$ opérations binaires.
   Conclusion : pour le calcul de $a_1.b_1$, on a $O((log(p))^2) + O((log(p))^2) = O(2(log(p))^2) = O((log(p))^2)$ opérations binaires. Pour le calcul modulo p, on effectue une division euclidienne de $a_1.b_1$ par p qui est de même complexité que la multiplication.
   On obtient donc une complexité en $O((log(p))^2 + (log(p))^2) = O((log(p))^2)$.

Proposition 2. *Soient $a, b \in \mathbb{N}^*$, $a \geq b$. Le calcul de pgcd(a,b) à l'aide de l'algorithme d'Euclide est de complexité $O((log(b))^3)$.*

La preuve va résulter du théorème suivant :

Théorème de Lamé. *Le nombre de divisions euclidiennes à effectuer pour trouver le pgcd de deux entiers naturels à l'aide de l'algorithme d'Euclide ne dépasse pas cinq fois le nombre de chiffres dans l'écriture en base 2 du plus petit des deux nombres.*

Démonstration. Pour prouver ce théorème, on va utiliser la suite de Fibonacci (F_i), définie par $F_1 = F_2 = 1$ et $F_{i+2} = F_{i+1} + F_i$ pour $i \geq 1$.
On définit le nombre d'or $\phi = \frac{1+\sqrt{5}}{2}$. Ce nombre est une solution de l'équation $x^2 - x - 1 = 0$.

Notons n le nombre de divisions qui apparaissent dans l'algorithme d'Euclide appliqué à a et b et posons $a = r_0$ et $b = r_1$. Alors l'algorithme d'Euclide s'écrit comme suivant:

$$(AE)\begin{cases} r_0 = r_1 q_1 + r_2 \\ r_1 = r_2 q_2 + r_3 \\ .... \\ r_{n-2} = r_{n-1} q_{n-1} + r_n \\ r_{n-1} = r_n q_n + 0 \end{cases} \qquad 0 < r_i < r_{i-1}\ ,\ 0 \leq i \leq n.$$

Montrons par récurence que $F_i \geq \phi^{i-2}$ pour tout $i \geq 2$. Il est clair que l'inégalité est vraie pour $i = 2$ et $i = 3$. Supposons l'inégalité vraie au rang $i-1$ et $i-2$, Alors

$$F_i = F_{i-1} + F_{i-2} \geq \phi^{i-3} + \phi^{i-4} = \phi^{i-4}(\phi + 1) = \phi^{i-4}\phi^2 = \phi^{i-2}.$$

L'inégalité est démontrée.

Montrons aussi par récurence sur k que, pour $0 \leq k \leq n$, $r_{n-k} \geq F_{k+2}$. On a $r_n \geq 1 = F_2$. Comme $q_n \geq 2$ (car sinon, on aura $(n-1)$ divisions dans l'algorithme) alors $r_{n-1} = r_n q_n \geq 1 \times 2 = F_3$. Supposons l'inégalité vraie au rang $k-1$ et $k-2$, Alors

$$r_{n-k} = r_{n+1-k} q_{n+1-k} + r_{n+2-k} \geq r_{n+1-k} + r_{n+2-k} \geq F_{k+1} + F_k = F_{k+2}.$$

L'inégalité est démontrée.

On a $b = r_1 = r_{n-(n-1)} \geq F_{n+1}$ et d'après la première inégalité $F_{n+1} \geq \phi^{n-1}$ donc $b \geq \phi^{n-1}$. D'autre part

$\ln(\phi) > \frac{1}{5}\ \ln(10) = \frac{1}{5}(\ \ln(2) + \ln(5)) > \frac{1}{5}\ \ln(2)$.
D'où

$\ln(b) \geq (n-1)\ln(\phi) > \frac{1}{5}(n-1)\ln(2)$.

b s'écrit en base 2 avec k chiffres, donc $b < 2^k$. Cela implique

$k\ \ln(2) > \ln(b) > \frac{1}{5}(n-1)\ln(2)$. Ainsi $n \leq 5k$. D'ou le résultat.

□

Preuve. *(proposition 2)*

On conclue d'après le théorème précédent que l'algorithme d'Euclide nécessite $O(log(b))$ divisions, chacune de complexité $O((log(b))^2)$ (on utilise ici la complexité de la multiplication et de l'addition qu'on a vu en proposition 1), donc la complexité totale est $O((log(b))^3)$.

Corollaire. $a^{-1} \pmod p$ *se calcule en* $O((log(p))^3)$.

En effet, on applique l'algorithme d'Euclide pour trouver $pgcd(a, p)$ et après on le remonte pour trouver l'inverse de a modulo p. Il nécessite donc $O(2(log(p))^3) = O((log(p))^3)$ opérations binaires.

3.3 Exponentiation rapide

Pendant le traitement de chiffrement ElGamal, on doit calculer $\alpha^k \pmod p$. Donc le problème qui se pose maintenant est de trouver une méthode efficace qui calcule rapidement α^k (on suppose ici $k \leq p-1$ sinon, on effectue la division euclidienne de k par $p-1$).
La méthode dont nous avons pris l'habitude est de calculer successivement $\alpha, \alpha^2, ..., \alpha^k$. On a donc effectué $k-1$ multiplications. Cette méthode peut être très longue si l'exposant est grand.
Une méthode efficace pour ce problème est l'exponentiation rapide.

Algorithme.

1. Ecrire k en base 2 : $k = \sum_{i=0}^{n} a_i 2^i$.
2. Calculer $\alpha^{2^i} \pmod p$, $0 \leq i \leq n$.
3. Calculer $\alpha^k \equiv \prod_{i=0}^{n} (\alpha^{2^i})^{a_i} \pmod p$.

Remarque.

1. Le choix de base 2 est dû au fait que l'entier a_i soit égal à 1 ou à 0 et cela minimise le temps du calcul.
2. Pour simplifier le calcul en étape 2, on utilise cette relation :

$$\alpha^{2^{i+1}} = (\alpha^{2^i})^2.$$

   Donc le nombre de multiplications de calcul de $\alpha^{2^{i+1}}$ sera égal au nombre de multiplications de calcul de α^{2^i} plus un ce qui est plus efficace.

Pour bien appliquer l'algorithme de l'exponentiation rapide au calcul de $\alpha^k \pmod p$ dans le chiffrement de ElGamal, nous présentons maintenant un exemple.

Exemple. Supposons $\alpha = 7$, $p = 41$ et $k = 27$.
On écrit tout d'abord 27 en base 2 : $27 = 2^4 + 2^3 + 2^1 + 2^0$.
Ensuite, on calcule $7^{2^i} \pmod{41}$:
$7^{2^0} = 7^1 \equiv 7 \pmod{41}$
$7^{2^1} = 7^2 = 49 \equiv 8 \pmod{41}$
$7^{2^2} = (7^2)^2 \equiv 8^2 \pmod{41} \equiv 23 \pmod{41}$
$7^{2^3} = (7^{2^2})^2 \equiv 23^2 \pmod{41} \equiv 37 \pmod{41}$
$7^{2^4} = (7^{2^3})^2 \equiv 37^2 \pmod{41} \equiv 16 \pmod{41}$
Enfin, on calcule :
$7^{27} \equiv 7 \times 8 \times 37 \times 16 \pmod{41} \equiv 24 \pmod{41}$.

Théorème 2. *Soit a un élément de $\mathbb{Z}/p\mathbb{Z}$ et soit m un entier naturel inferieur à $p-1$. Alors le calcul de $a^m \pmod p$ par l'algorithme précédent a pour complexité $O((log(p))^3)$.*

Démonstration. Cet algorithme nécessite $O(log(m))$ multiplications modulaires et donc $O(log(p))$ car $m \leq p-1$. Chaque multiplication est de complexité $O((log(p))^2)$. Comme on a $O((log(p))^2)$ opérations binaires pour le calcul modulo p, alors la complexité totale est $O((log(p))^3 + (log(p))^2) = O((log(p))^3)$. □

3.4 Complexité ElGamal

Dans cette section, on va utiliser tous les proprietés qu'on a vu sur la complexité.

Pour le chiffrement de ElGamel , on calcule $\xi_1 \equiv \alpha^k \pmod p$ qui est de complexité $O((log(p))^3)$. Puis, on calcule aussi $\xi_{2,i} \equiv n_i\beta^k \pmod p$ qui est de complexité $O((log(p))^2 + (log(p))^3)$.
La complexité totale sur un bloc est donc

$$O((log(p))^2 + 2(log(p))^3) = O((log(p))^3).$$

Pour le déchiffrement, on calcule d'abord $\xi_1^a \pmod p$ (ξ_1 et ξ_2 sont des donnés). La complexité vaut $O((log(p))^3)$. Ensuite, on calcule l'inverse du résultat trouvé. On a vu que ce inverse se calcule avec complexité $O((log(p))^3)$. Donc, pour le calcul de $(\xi_1^a)^{-1} \pmod p$, on a

$$O((log(p))^3 + (log(p))^3) = O((log(p))^3)$$

pour complexité.
Finalement, la complexité de $\xi_{2,i}({\xi_1}^a)^{-1} \pmod p$ est

$$O((log(p))^3 + (log(p))^2) = O((log(p))^3).$$

La complexité du déchiffrement sur un bloc est donc $O((log(p))^3)$.

4 Calcul du logarithme discret

On rappelle que le problème du logarithme discret consiste à trouver l'exposant a tel que $\alpha^a \equiv \beta \pmod p$ (α et β donnés).
Voici un algorithme dû à Shanks (Pas de bébé, pas de géant) qui résout ce problème de façon plus efficace que le test de toutes les possibilités.

4.1 Algorithme de Shanks

Soit $m = \lfloor\sqrt{p-1}\rfloor + 1$.
1.Calculer $\alpha^{mi} \pmod p$, pour $0 \leq i \leq m-1$.
2.Trier la liste $L_1 = \{(i, \alpha^{mi}), 0 \leq i \leq m-1\}$ selon un ordre sur le second membre.
3.Calculer $\beta\alpha^{-j} \pmod p$, pour $0 \leq j \leq m-1$.

4.Trier la liste $L_2 = \{(j, \beta\alpha^{-j}), 0 \leq j \leq m-1\}$ selon le même ordre que L_1.
5.Trouver i_0 et j_0 qui vérifient $\alpha^{mi_0} = \beta\alpha^{-j_0}$.
6.Définir la solution $log_\alpha(\beta) \equiv\ mi_0 + j_0 \pmod{p-1}$.

Justification. Le terme commun aux deux listes existe toujours.
En effet, soit $\beta \in (\mathbb{Z}/p\mathbb{Z})^*$. Alors il existe a, $0 \leq a \leq p-1$, tel que $\alpha^a \equiv\ \beta \pmod{p}$. Effectuant la division euclidienne de a par m: $a = qm + r$, $0 \leq r \leq m-1$. On a $a \geq qm$ car $r \geq 0$. De même $a < m^2$ car $a \leq p-1 < m^2$. Donc cela implique que $q \leq m-1$. D'ou l'existence de $i_0 = q$ et $j_0 = r$, $0 \leq i_0, j_0 \leq m-1$, tel que $\beta \equiv\ \alpha^{j_0+mi_0} \pmod{p}$ *ie* $\beta\alpha^{-j_0} \equiv\ \alpha^{mi_0} \pmod{p}$.

Exemple. Soient $p = 113$, $\alpha = 3$ et $\beta = 2$.

Alors

$$m = 11 \quad \text{et} \quad \alpha^{11} \pmod{113} = 76.$$

On calcule tout d'abord $76^i \pmod{113}$ pour $0 \leq i \leq 10$, puis on trie la liste L_1 suivant l'ordre croissant de second membre. On obtient

$$L_1 = \{(2,13), (9,19), (6,50), (4,56), (7,71),$$

$$(5,75), (1,76), (3,84), (8,85), (10,88)\}.$$

Pour la seconde liste, on calcule pour $0 \leq j \leq 10$, $2 \times 3^{-j} \pmod{113}$ ensuite, on trie la liste L_2 selon le même ordre que L_1. on a donc

$$L_2 = \{(10,9), (4,7), (7,17), (3,21), (9,27),$$

$$(5,40), (6,51), (2,63), (1,76), (8,81)\}.$$

On remarque que la deuxième coordonnée du deux couples $(1,7)$ et $(1,7)$ est le même. Donc on a

$$3^{11} = 2 \times 3^{-1}.$$

Alors $log_3(2) = 11 + 1 = 12$.

4.2 Complexité de Shanks

La complexité de l'algorithme de Shanks est $O(\sqrt{p}.(log(p))^3)$. En effet,

Pour i fixé, le calcul de $\alpha^{mi} \pmod{p}$ se fait en $O((log(p))^3)$ opérations binaires. Donc la complexité de la première étape est $O(\sqrt{p}.(log(p))^3)$. Pour le trie des listes, on a $O(\sqrt{p}.log(p))$ opérations binaires. La troisième étape est de complexité $O(\sqrt{p}.(log(p))^3)$ car pour j donné, la complexité du calcul de $\beta.\alpha^{-j} \pmod{p}$ est

$$O((log(p)^3 + (log(p))^3 + (log(p))^2) = O((log(p))^3).$$

Finalement, la complexité de Shanks est

$$O(\sqrt{p}.(log(p))^3 + \sqrt{p}.(log(p))^3 + \sqrt{p}.(log(p))) = O(\sqrt{p}.(log(p))^3).$$

Remerciements.

Je tiens à remercier mon encadreur Michaël Bulois pour m'avoir donner l'occasion de travailler avec lui. Je tiens particulierement à le remercier pour sa patience, sa persévérence et tous le temps qu'il m'a consacré ainsi que pour toutes les réponses à mes questions. Cette experience fut très enrichissante car j'ai découvert la notion de cryptographie et l'initiation à la recherche. Je remercie également M Driss Essouabri d'avoir lu mon rapport de TER.
Ce fut une experience inoubliable.

References

[Sti] Douglas Stinson, *Cryptographie, Théorie et pratique*, Vuibert 2003.

[Ste] Marie-Aude Steineur, *Etude de la Primalité motivée par le besoin de Nombres Premiers dans le Chiffrement RSA*, mémoire de magistère accessible à l'adresse http://www-magistere.u-strasbg..fr/spip.php?article127

[Chr] Chabot Christophe, *Arithmétique modulaire pour la cryptographie* accessible à l'adresse http://www.unilim.fr/pages_perso/christophe.chabot/doc/cours3.pdf

[Bru] Martin Bruno, *Cryptographie à clé publique II* accessible à l'adresse http://deptinfo.unice.fr/ ~bmartin/5-CS-4.pdf

[Ghi] Labouret Ghislaine, *Introduction à la cryptographie*, Hervé Schauer Consultants accessible à l'adresse http://www.hsc.fr/ressources/cours/crypto/crypto.pdf

[Dan] Mercier Dany-Jack, *Coût de l'algorithme d'Euclide et CAPES interne 2000*, 24 janvier 2003 accessible à l'adresse http://megamaths.perso.neuf.fr/themes/calg0002.pdf

[Nic] Rémond Nicolas - Gury Steve, *Cryptologie : El Gamal d'après Diffie-Hellman* accessible à l'adresse http://s.gury.free.fr/Crypto/elgamal.pdf