Cryptographie à clé publique et factorisation

Chantal David
Université Concordia

Collège Edouard-Montpetit
Février 2003

Cryptosystèmes

Un système de cryptographie est un tuple (K, e_K, d_K) où

- K est la clé d'encodage
- e_K est la fonction d'encodage
- d_K est la fonction de décodage

et tel que pour tout message m on a

$$d_K(e_K(m)) = m.$$

Exemple:

La clé d'encodage est la permutation suivante de l'alphabet:

A	B	C	D	E	F	G	H	I	J	K	L	M
Z	P	V	A	C	K	X	U	W	Y	I	F	H

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	R	N	G	D	T	M	J	Q	E	L	O	S

On a donc,

$$\begin{aligned} e_K(BONJOUR) &= PRBYRJD \\ d_K(ACHZWB) &= DEMAIN \\ d_K(PRBYRJD) &= BONJOUR \end{aligned}$$

et en particulier

$$\begin{aligned} d_K(e_K(BONJOUR)) &= d_K(PRBYRJD) \\ &= BONJOUR. \end{aligned}$$

Cryptographie classique et à clé publique

Evidemment, le message est très facile à décoder. Les attaques possibles sont:

- analyses statistiques;

- interception de la clé d'encodage.

Dans tous les systèmes de cryptographie classique, si on connait la clé d'encodage, on connait la clé de décodage. En d'autres termes, on peut calculer d_K à partir de e_K. Et pour que Alice et Bruno puissent s'échanger des messages codés, il doivent s'échanger secrètement la clé d'encodage et de décodage K.

Dans les systéme de cryptographie à clé publique, on "ne peut pas" calculer d_K à partir de e_K. La clé d'encodage peut donc être publique, et la deuxième attaque est impossible. Ceci est particulièrement adapté aux communications sur Internet.

Cryptographie à clé publique

On "ne peut pas" calculer d_K à partir de e_K, car il faudrait résoudre un problème intraitable (les calculs ne peuvent s'effectuer en temps raisonnable).

Quelques cryptosystèmes à clé publique:

1. RSA (Rivest-Shamir-Adelman, 1978)
Ce système est basé sur le problème de la factorisation des entiers.

2. El-Gamal (1985)
Ce système est basé sur le problème du logarithme discret.

3. Courbes elliptiques (Koblitz, 1987)
Ce système est basé sur le problème du logarithme discret dans le groupe des points d'une courbe elliptique sur un corps fini.

Théorie de la complexité

Soit T le nombre d'opérations élémentaires d'un algorithme en fonction de la taille du problème.

Remarque: La taille d'un entier n est son nombre de décimales (dans une base donnée).

Exemple: 1295 possède 4 décimales en base 10, et 11 décimales en base 2.

$$\begin{aligned} 1245 &= 1024 + 128 + 64 + 16 + 8 + 4 + 1 \\ &= (10011011101)_2 \end{aligned}$$

En général, le nombre de décimales de n en base b est

$$[\log_b(n)] + 1.$$

Par la loi des logarithmes, le nombre de décimales dans 2 bases différentes ne change que par une constante. On écrit donc

$$\text{Taille de } n \leq C \log n = O(\log n).$$

Soit T le nombre d'opérations élémentaires d'un algorithme en fonction de la taille du problème.

Exemple:

$$\begin{aligned} T(n+n) &= O(\log n); \\ T(n*n) &= O\left(\log^2 n\right); \\ T(n/a) &= O(\log a \log n) = O\left(\log^2 n\right). \end{aligned}$$

Dans le dernier exemple, diviser n par a ($a \leq n$) veut dire trouver q et r tels que

$$n = aq + r, \qquad 0 \leq r < a.$$

Définition Un algorithme est polynômial si son nombre d'opérations élémentaires est $O(\log^d n)$ pour un certain entier positif d (i.e. c'est un polynôme de degré d dans la taille de n).

En pratique,
polynômial $\leftrightarrow$ traitable
non-polynômial $\leftrightarrow$ non-traitable.

Factorisation

Définition Un entier positif n est premier si ses seuls diviseurs sont 1 et n.

Il y a une infinité de premiers (Euclide), qui sont

$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, \ldots$

Théorème fondamental de l'arithmétique Tout entier positif s'écrit de façon unique comme un produit de premiers.

Exemples (a) $12 = 2^2 \cdot 3$;
(b) $354363 = 3 \cdot 41 \cdot 43 \cdot 67$;
(c) $146398 = 2 \cdot 7 \cdot 10457$.

On veut évaluer

$$T(\text{factoriser } n),$$

le nombre d'opérations élémentaires pour factoriser un entier positif n.

Algorithmes de factorisation

1. Algorithme trivial

Pour chaque entier $1 \leq k \leq n$, on vérifie si k divise n. On a donc

$$T(\text{factoriser } n) = O(n \log^2 n)$$

ou

$$T(\text{factoriser } n) = O(\sqrt{n} \log^2 n)$$

car on n'a besoin que de $1 \leq k \leq \sqrt{n}$.
Cet algorithme n'est pas polynômial.

2. Meilleurs algorithmes connus

Ce sont le crible quadratique et le crible des corps de nombres. Leur temps n'est pas polynômial (en $\log n$), mais on a

$$T_{CCN}(\text{factoriser } n) = O(n^{\epsilon})$$

pour tout $\epsilon > 0$. On les appelle des algorithmes sous-exponentiels.

RSA-155

L'entier suivant a récemment été factorisé en utilisant ces algorithmes de factorisation et des années de temps CPU sur des ordinateurs à travers le monde...

$$
\begin{aligned}
&1094173864157052742180970732204 \\
&0357612003732945449205909138421 \\
&3147634998428893478471799725789126 \\
&7332497625752899781833797076537244 0 \\
&2714674353159335433389 7 \\
&= \\
&1026395928297411057720541 9 \\
&6573991675900716567808038066 \\
&8033419335217907113077 79 \\
&\times \\
&1066034883800168454820927220 \\
&3600112878679207958575989291522 2 \\
&7060823719306280864 3
\end{aligned}
$$

On peut trouver sur la page WEB
http://www.rsasecurity.com/rsalabs/challenges/factoring/numbers.html
d'autres défis de factorisation:
RSA-576 ($10,000),
RSA-2048 ($200,000).

3. Algorithme polynômial

Existe-t-il un algorithme polynômial pour factoriser n? Les experts croient que non, mais personne ne peut le prouver. La sécurité de RSA est basée sur le fait qu'il n'existe pas d'algorithme polynômial pour factoriser n.

RSA Alice choisit 2 grands premiers p et q, et calcule $n = pq$.

Clé d'encodage publique: $K_E = n$.
Clé de décodage privée: $K_D = \{p, q\}$.

Pour envoyer un message codé à Alice, Bruno utilise la clé publique $K_E = n$. Pour décoder le message de Bruno, il faut soit connaitre $K_D = \{p, q\}$ d'avance, soit factoriser n et ainsi retouver p et q.

Arithmétique modulaire

Soit n un entier positif. On définit

$$\begin{aligned} a \equiv b \bmod n &\iff n \text{ divise } b-a \\ &\iff a \text{ et } b \text{ ont le même reste} \\ &\quad\ \ \text{lorsque divisés par } n \end{aligned}$$

Soit $\mathbb{Z}/n\mathbb{Z}$ l'ensemble des classes d'équivalence modulo n. Alors

$$\mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, 3, \ldots n-1\}$$

Exemple: Dans $\mathbb{Z}/7\mathbb{Z} = \{0, 1, 2, 3, 4, 5, 6\}$, on calcule

$$\begin{aligned} 2+3 &= 5 \\ 6+5 &= 11 \equiv 4 \bmod 7 \\ 3+4 &= 7 \equiv 0 \bmod 7 \\ 5 \cdot 6 &= 30 \equiv 2 \bmod 7 \\ 2 \cdot 4 &= 8 \equiv 1 \bmod 7 \end{aligned}$$

Nous sommes surtout interessés par la structure multiplicative des éléments de $\mathbb{Z}/n\mathbb{Z}$. En particulier, quels sont les éléments de $\mathbb{Z}/n\mathbb{Z}$ qui ont un inverse multiplicatif?

Calculer les inverses modulo 7 et 4:

a mod 7	0	1	2	3	4	5	6
a^{-1} mod 7	-	1	4	5	2	3	6

a mod 4	0	1	2	3
a^{-1} mod 4	-	1	-	3

Définition: Soit $(\mathbb{Z}/n\mathbb{Z})^*$ l'ensemble des éléments inversibles de $(\mathbb{Z}/n\mathbb{Z})$.

Question: Déterminer $(\mathbb{Z}/n\mathbb{Z})^*$.

Définition: Soit $(\mathbb{Z}/n\mathbb{Z})^*$ l'ensemble des éléments inversibles de $(\mathbb{Z}/n\mathbb{Z})$.

Théorème:

$$(\mathbb{Z}/n\mathbb{Z})^* = \{1 \leq a \leq n \mid \text{pgcd}(a,n) = 1\}\,.$$

Définition:

$$\begin{aligned} \phi(n) &= \#(\mathbb{Z}/n\mathbb{Z})^* \\ &= \#\{1 \leq a \leq n \mid \text{pgcd}(a,n) = 1\} \end{aligned}$$

Exemples:
(a) $\phi(7) = 6$;
(b) $\phi(4) = 2$;
(b) $\phi(p) = p - 1$, p premier;
(b) $\phi(pq) = (p-1)(q-1)$, p, q premiers.

Ordre des éléments

Définition: Soit $a \in (\mathbb{Z}/n\mathbb{Z})^*$. L'ordre de a est le plus petit entier positif k tel que

$$a^k \equiv 1 \bmod n.$$

Exercice: Déterminez l'ordre des éléments de $(\mathbb{Z}/11\mathbb{Z})^*$.

a	1	2	3	4	5	6	7	8	9	10
k	1	10	5	5	5	10	10	10	5	2

Question: Qu'observez-vous à propos de l'ordre des éléments de $(\mathbb{Z}/11\mathbb{Z})^*$?
Indice: Comparez avec $\phi(11) = \#(\mathbb{Z}/11\mathbb{Z})^*$.

Question: Qu'observez-vous à propos de l'ordre des éléments de $(\mathbb{Z}/11\mathbb{Z})^*$?

Les éléments de $(\mathbb{Z}/11\mathbb{Z})^*$ ont ordre $1, 2, 5, 10$. Ce sont tous des diviseurs de

$$10 = \phi(11) = \#(\mathbb{Z}/11\mathbb{Z})^*.$$

Ceci est vrai en général.

Petit Théorème de Fermat: Soit n un entier positif, et a un entier tel que $1 \leq a \leq n$ et $\text{pgcd}(a, n) = 1$. Alors

$$a^{\phi(n)} \equiv 1 \text{ mod } n.$$

Preuve: Soit k l'ordre de a dans $(\mathbb{Z}/n\mathbb{Z})^*$. Alors, k divise $\phi(n) \iff \phi(n) = kd$ pour un entier d. On a donc

$$\begin{aligned} a^{\phi(n)} &= a^{kd} = (a^k)^d \\ &\equiv 1^d = 1 \text{ mod } n. \end{aligned}$$

Applications du Petit Théorème de Fermat:
- Tests de primalité
- Cryptosystème RSA

Tests de Primalité

Si $n = p$ un nombre premier, alors le Petit Théorème de Fermat devient

$$a^{p-1} \equiv 1 \mod p$$

pour $1 \leq a \leq p-1$.

On peut se poser la question inverse:
Si n est un entier tel que

$$a^{n-1} \equiv 1 \mod n$$

pour un certain nombre de a, est-ce que n est premier?

C'était la base de tous les tests de primalité (avant août 2002!). On obtient ainsi

- un algorithme polynômial sous l'hypothèse de Riemann généralisée
- un algorithme probabiliste polynômial

pour tester si un entier n est premier.

Tests de Primalité

Théorème de Agrawal-Kayal-Saxena (Août 2002) Il existe un algorithme polynômial pour tester si un entier n est premier ou composé. Plus précisement,

$$T_{AKS}(n \text{ est premier}) = O\left(\log^{12}(n)\right).$$

Cet algorithme est élémentaire, et est basé sur une toute autre méthode de détection des premiers que celle du Petit Théorème de Fermat.

Remarque: Un algorithme polynômial pour tester si n est premier n'est d'aucune utilité pour factoriser $n = pq$ en temps polynômial !

Cryptosystème RSA

Alice trouve 2 grands premiers p et q;
Alice calcule $n = pq$;
Alice calcule $\phi(n) = (p-1)(q-1)$;
Alice choisit un entier e entre 1 et n tel que le pgcd de e et $\phi(n)$ est 1;
Alice calcule d entre 1 et n tel que $ed \equiv 1 \bmod n$.

Toutes ces opérations peuvent être faites en temps polynômial.

Clé d'encodage publique: n, e

Clé de décodage privée: d (ou $\phi(n)$, ou p, q).

Soit un entier $1 \leq m \leq n$ qui est le message que Bruno veut coder. Bruno calcule

$$e_K(m) = m' \equiv m^e \bmod n.$$

Pour retouver m a partir de $e_K(m)$, Alice calcule

$$d_K(m') = (m')^d \bmod n.$$

Cryptosystème RSA

On a vu que

$$\begin{aligned} e_K(m) &= m^e \bmod n \\ d_K(m') &= (m')^d \bmod n \end{aligned}$$

Il faut montrer que c'est un cryptosystème, i.e.

$$d_K(e_K(m)) \equiv m \bmod n.$$

Preuve:

$$\begin{aligned} d_K(e_K(m)) &= d_K(m^e) \equiv (m^e)^d \bmod n \\ &= m^{ed} \bmod n \\ &= m^{1+k\phi(n)} \bmod n \end{aligned}$$

car $ed \equiv 1 \bmod \phi(n) \Leftrightarrow ed = 1 + k\phi(n)$ pour un certain entier k.

On a donc

$$\begin{aligned} d_K(e_K(m)) &\equiv m^{1+k\phi(n)} \bmod n \\ &\equiv m(m^{\phi(n)})^k \bmod n \\ &\equiv m\ 1 \bmod n = m \bmod n \end{aligned}$$

par le Petit Théoreme de Fermat.

Message encrypté avec RSA

Voici un exemple d'un message codé avec le système de cryptographie à clé publique RSA. Les clés d'Alice sont les suivantes

Clé publique:

$$
\begin{aligned}
n &= 350714609064434377778725 \\
&\quad 726487501785999 \\
e &= 26479
\end{aligned}
$$

Clé privée:

$$
\begin{aligned}
p &= 5307594994995497073 \\
q &= 66077820885729742448 63 \\
\phi(n) &= 350714609064434371170890561 9 \\
&\quad 19532044064 \\
d &= 154556016963362841391031457647 \\
&\quad 155082223
\end{aligned}
$$

On utilise l'alphabet suivant:

$$. = 0,\ A = 1,\ B = 2,\ C = 3,\ D = 4,\ E = 5,$$
$$F = 6,\ G = 7,\ H = 8\ I = 9\ J = 10,\ K = 11,$$
$$L = 12,\ M = 13,\ N = 14,\ O = 15,\ P = 16,$$
$$Q = 17,\ R = 18,\ S = 29,\ T = 20,\ U = 21,$$
$$V = 22,\ W = 23,\ X = 24,\ Y = 25,\ Z = 26$$

Il y a donc 27 lettres dans l'alphabet, et comme $[\log_{27}(n)] = 26$, les messages sont envoyés par blocs de 26 lettres. Bruno veut encrypter le message

HERE.IS.AN.EXAMPLE.FOR.YOU

En utilisant les équivalences numériques pour les lettres de l'alphabet, le message devient

8 5 18 5 0 9 19 0 1 14 0 5 24 1 13 16
12 5 0 6 15 18 0 25 15 21

qui est un nombre de 26 chiffres en base 27.

Bruno transforme ce nombre en base 10, et obtient l'entier $1 \leq m \leq n$ suivant

$$\begin{aligned} m &= 499395015958794078593992 \\ &\quad 8670834544841 \end{aligned}$$

qui est la valeur numérique de son message à encrypter. Bruno calcule maintenant

$$m' \equiv m^e \bmod n,$$

et obtient l'entier $1 \leq m' \leq n$ suivant

$$\begin{aligned} m' &= 182014418498730363326322 \\ &\quad 89850319898454 7 \end{aligned}$$

Puis, il retransforme cet entier de la base 10 à la base 27, ce qui donne

$$\begin{array}{c} 11\ 2\ 6\ 8\ 26\ 0\ 0\ 7\ 14\ 0\ 22\ 8\ 9\ 16\ 22 \\ 16\ 26\ 15\ 14\ 21\ 10\ 6\ 12\ 26\ 4\ 12\ 16 \end{array}$$

et en utilisant les équivalences numériques pour les lettres de notre alphabet, le message encrypté est

$$KBFHZ..GN.VHIPVPZONUJFLZDLP$$