Cryptographie à clé publique

Bruno MARTIN,
Université Côte d'Azur

Cryptographie à clé publique

1976 : Invention de Diffie et Hellman [2] ; 1re phrase prophétique :

> *We stand today on the brink of a revolution in cryptography.*

Idée géniale : asymétrique ; chiffrement $\neq$ du déchiffrement.

- chiffrement par clé **publique** (pk)
- déchiffrement avec clé **privée** (sk)

Utile pour distribuer les clés : publiées dans un annuaire
Le principe de Kerckhoff (1883) est d'autant plus d'actualité

> *La sécurité d'un chiffre ne doit pas dépendre du secret de l'algorithme mais seulement du secret de la clé.*

Fonction à sens unique

Soit M et C deux ensembles et $f : M \to C$. $f(M)$: image de l'ensemble M par f. f est **à sens unique** (OW) si

- pour tout x de M, il est facile de calculer $f(x)$ (f calculable en temps polynomial) et
- il est difficile de trouver, pour la plupart des $y \in f(M)$ un $x \in M$ tel que $f(x) = y$ (problème « difficile » [4, 5, 1]).

Mais, s'il est difficile de trouver, pour la plupart des $y \in f(M)$ un $x \in M$ tel que $f(x) = y$, déchiffrer aussi difficile que cryptanalyser ! Autre notion nécessaire pour permettre le déchiffrement et rendre la cryptanalyse aussi difficile que possible.

$\to$ Notion de trappe.

Fonction à sens unique à trappe

$f : M \to C$ à sens unique est **à trappe** si le calcul dans le sens inverse est efficace en connaissant la trappe (secrète).

La trappe permet de construire une fonction g telle que $g \circ f = Id$. Calcul dans le sens direct est facile et dans le sens inverse, calculatoirement impossible sans connaître g.

- construire des couples (f, g) doit être facile
- publier f ne doit rien révéler sur g

C'est là qu'on retrouve formalisée l'idée d'utiliser deux algorithmes différents, un pour chiffrer f et un pour déchiffrer g.

Définition PKC

Définition

Un chiffre à clé publique est un triplet PPT Π *=(Gen, Enc, Dec) :*

- *Gen* $:1^n \to (pk, sk)$ *resp. clé publique et privée tq* $|pk| \approx |sk| \approx n$.
- *Enc* $: c \leftarrow Enc_{pk}(m)$ *où* $m \in M(pk)$
- *Dec* $: m := Dec_{sk}(c)$ *(déterministe)*

On demande que $Dec_{sk}(Enc_{pk}(m)) = m$ *pour tous les* $m \in M$ *et toutes les clés* (pk, sk) *produites par Gen.*

Formalisation OW

Définition (OW)

$f : \{0,1\}^\star \to \{0,1\}^\star$ *est à sens unique si :*

- *il existe un algo en temps polynomial pour calculer* $f(x)$
- *pour tout adversaire A PPT, il existe negl. tq :*

$$Pr[Invert_{A,f}(n) = 1] \leq negl(n)$$

$Invert_{A,f}(n)$ formalise l'attaque contre OW f :

1. $m \xleftarrow{u} \{0,1\}^n$; $c := f(m)$
2. A reçoit 1^n et c et renvoie m'
3. A réussit (i.e. renvoie 1) ssi $f(m') = c$

Sécurité calculatoire : exemple factorisation

Cryptanalyste doit faire plus de calculs que la durée de vie du clair.

- Donné $n = pq$
- trouver p (et q).

Année	$\log_2(\sharp Op.)$	$\log_2(n)$	1 MIPS (années)	500 MIPS (années)
< 2000	64	768	2^{19}	2^{10}
< 2010	80	1024	2^{35}	2^{26}
< 2020	112	2048	2^{67}	2^{58}
< 2030	128	3072	2^{83}	2^{74}

NB : 2^{80} opérations correspond à 2^{35} ans (1 MIPS). Pour info, un i7 développe au max 318 MIPS et certains i9 4000 MIPS.
Défis de factorisation jusqu'à 2009 jusqu'à RSA-768 (232 chiffres).

Sécurité prouvée : preuves par réduction

- Hypothèse : le problème algorithmique Π est difficile (pas algo poly. sauf...) avec Π=RSA, DLP, DDH, CDH,...
- Réduction :
  - si un adversaire A PPT casse le chiffre
  - alors on peut utiliser A pour résoudre Π en temps poly.
- résultat de sécurité : il n'existe pas d'adversaire poly.

Expérience IND-CPA : $\text{PubK}^{\text{CPA}}_{A,\Pi}$

1. $\text{Gen}(1^n)$ produit (pk, sk)
2. l'adversaire A reçoit pk et un accès à (l'oracle) $\text{Enc}_{pk}(.)$. A retourne $m_0, m_1 \in M(pk)$ de même long.
3. $b \leftarrow \{0,1\}$; calculer $c \leftarrow Enc_{pk}(m_b)$ et envoyer c à A : le défi chiffré
4. A a toujours accès à $\text{Enc}_{pk}(.)$. et retourne un bit b'
5. A réussit l'expérience (i.e. renvoie 1) ssi $b = b'$

IND-CPA exprime le fait qu'il est impossible de distinguer quel message a été chiffré et souligne le fait qu'il faut un chiffrement probabiliste !

Expérience IND-EAV : $\text{PubK}^{\text{EAV}}_{A,\Pi}$

1. $\text{Gen}(1^n)$ produit (pk, sk)
2. l'adversaire A reçoit pk ~~et un accès à (l'oracle) $\text{Enc}_{pk}(.)$~~. A retourne $m_0, m_1 \in M(pk)$ de même long.
3. $b \leftarrow \{0,1\}$; calculer $c \leftarrow Enc_{pk}(m_b)$ et envoyer c à A : le défi chiffré
4. A ~~a toujours accès à $\text{Enc}_{pk}(.)$. et~~ retourne un bit b'
5. A réussit l'expérience (i.e. renvoie 1) ssi $b = b'$

qui est équivalente à $\text{PubK}^{\text{CPA}}_{A,\Pi}$

Définition de la sécurité

Définition

Π est IND-CPA si, pour tout adversaire A PPT, il existe $negl(.)$ tq :

$$Pr(PubK^{CPA}_{A,\Pi}(n) = 1) \leq \frac{1}{2} + negl(n)$$

L'oracle de chiffrement n'est pas nécessaire, donc si Π IND-EAV, alors Π IND-CPA.

Le premier chiffre à clé publique

Inventé en 1978 par Rivest Shamir et Adleman.
Ils cherchaient une contradiction au concept de clé publique.
Après beaucoup d'efforts, ils sont finalement parvenus au résultat inverse et ont été récompensés du Turing Award en 2002 !
http://www.acm.org/awards/turing_citations/rivest-shamir-adleman.html

RSA

Rappels mathématiques

Indicatrice d'Euler d'un entier n notée $\varphi(n)$: nombre d'entiers compris entre 1 et n premiers avec n. On a $\varphi(1) = 1$ et pour p premier, $\varphi(p) = p - 1$.

$$\varphi(n) = \text{card}\{j \in \{1, \ldots, n\} : \gcd(j, n) = 1\}$$

Calcul : décomposer $n = \prod_{p|n, p \text{ premier}} p^{\alpha_p}$ alors,
$\varphi(n) = \prod_{p|n, p\text{premier}}(p^{\alpha_p} - p^{\alpha_p - 1}) = n \prod_{p|n}(1 - \frac{1}{p})$.

Exemple

$\varphi(12) = (4-2)(3-1) = 12(1-\frac{1}{2})(1-\frac{1}{3}) = 4$

Le chiffre de Rivest Shamir et Adleman (1978)

Repose sur la difficulté calculatoire de factoriser un nombre **et** sur la difficulté calculatoire de dire si un nombre est premier.
Par exemple, 1829 est-il premier ?

Non : on vous donne 31 et 59, on vérifie en les multipliant que 1829 est leur produit, mais les trouver est beaucoup plus difficile. Surtout qu'on ne connait pas a priori le nombre de facteurs premiers de 1829.

Ou bien, 7919 est-il composé ?

Non, mais le certificat de primalité est plus ≪ difficile ≫ à exhiber.

Petit théorème de Fermat - Euler

On utilise le théorème d'Euler :

Théorème (Fermat-Euler)

$$m^{\varphi(n)} \equiv 1 \mod n \text{ si } \gcd(m, n) = 1$$

qui généralise le petit théorème de Fermat :
Pour p premier, tout entier m vérifie :

$$m^p \equiv m \mod p$$

et si $p \not| m$,

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

Théorème Chinois des restes

Théorème

Soit $n_1, \ldots, n_k$ et $x_1, \ldots, x_k$ entiers tq pour toute paire (i, j), on a

$$x_i \equiv x_j \mod \gcd(n_i, n_j)$$

Il existe $x \in \mathbb{N}$ tq $x \equiv x_i \mod n_i$ pour $1 \leq i \leq k$. De plus, x est unique modulo le ppcm de $n_1, \ldots, n_k$.

Corollaire

Soit $n_1, \ldots, n_k$ premiers 2 à 2. Alors, pour tout entier x_i, il existe un unique entier x modulo $\prod n_i$ tq

$$x \equiv x_i \mod n_i \qquad 1 \leq i \leq k$$

Algorithme de calcul

Input : $x_i, n_i \quad 1 \leq i \leq n$
$n = \prod_{i=1}^{k} n_i$
for $i = 1$ à k **do**
calculer n/n_i
$\gcd(n/n_i, n_i) =$
$= s_i \frac{n}{n_i} + t_i n_i = 1$
$c_i = x_i s_i \mod n_i$
end for
Output : $\sum_{i=1}^{k} c_i \frac{n}{n_i}$

i	1	2	3	
x_i	2	3	2	
n_i	3	5	7	$n = 105$
$\frac{n}{n_i}$	35	21	15	
gcd	-1.35	1.21	1.15	
	12.3	-4.5	-2.27	
$s_i x_i$	-2	3	2	
mod	-2	3	2	
$c_i \frac{n}{n_i}$	-70	63	30	$x = 23$

Qiu JiuShao : combien l'armée de Han Xing a-t-elle de soldats si, rangés en 3 colonnes, il en reste 2, par 5 colonnes, il en reste 3 et par 7 colonnes, il en reste 2 ?

Factorisation et primalité : Crible d'Eratosthène

Problème

INSTANCE : *Un entier n*
QUESTION : *n est-il premier ?*

Problème algorithmique par excellence. Très étudié.
Nombreux critères permettent de vérifier si un entier n est premier.
On divise n par tous les entiers impairs compris entre 3 et $\lfloor \sqrt{n} \rfloor$.
Méthode connue depuis l'antiquité et efficace pour $n < 10^{12}$.
Crible d'Eratosthène en $O(\sqrt{n})$. Pas polynomial ! Complexité en temps n'est pas polynôme en la longueur de l'entrée, en $\log(n)$.
Il est **pseudo polynomial**.

Test de Lucas

Définition

*Soit $a \in \mathbb{Z}$ et $n \in \mathbb{N}$ tq $\gcd(a, n) = 1$, l'***ordre multiplicatif** *de a $\mod n$ est le plus petit entier k tq $a^k \equiv 1 \mod n$ (ou $a^k - 1 | n$).*

Théorème

n est premier ssi $\exists g \in \mathbb{Z}_n^\star$ tel que $g^{n-1} \equiv 1 \mod n$ et $g^{\frac{n-1}{p}} \not\equiv 1 \mod n$ pour tout p facteur premier de $n - 1$.

Preuve [6] pour n premier, g primitif modulo n remplit la condition. Réciproquement, si un tel g existe, alors l'ordre de g dans $\mathbb{Z}_n^\star$ est forcément $n - 1$. Or, tout élément de $\mathbb{Z}_n^\star$ est d'ordre un diviseur de $\varphi(n)$; comme $\varphi(n) \leq n - 1$, on a $\varphi(n) = n - 1$, équivalent à n premier. □

On en déduit le test de primalité de Lucas. Il permet de construire un algorithme non-déterministe en temps polynomial. Primalité $\in$NP .

Et le complémentaire de Primalité ?

La reconnaissance des nombres composés est dans NP ; construire p.e. un algorithme de type deviner/vérifier pour leur factorisation.
Plus précisément, on considère le problème FACTM des facteurs majorés.

Problème

INSTANCE : *Un entier n et un nombre $M \leq n$.*
QUESTION : *Existe-t-il un diviseur de n inférieur à M ?*

La difficulté de factoriser équivaut à la difficulté de résoudre FACTM. En effet, si FACTM$\in$ P, n se factorise en temps poly. par dichotomie : on cherche s'il y a un facteur de n dans l'intervalle $[1, \sqrt{n}]$ en résolvant FACTM avec $M = \lfloor \sqrt{n} \rfloor$; s'il n'y en a pas, n est premier. S'il y en a, on cherche si n a un facteur dans $[1, \sqrt{n}/2]$, sinon on cherche s'il a un facteur dans l'intervalle $[\sqrt{n}/2, \sqrt{n}]$ etc. En conséquence,

Théorème (Pratt)

PRIMALITÉ$\in$ NP $\cap$ co-NP .

Primes $\in$ P, Agrawal, Kayal et Saxena (2002)

A partir de l'identité :

$$\text{si } \gcd(a,p) = 1, \qquad [(x-a)^p \equiv (x^p - a) \mod p \Leftrightarrow p \text{ premier}]$$

AKS ont construit un **algo. dét. de test de primalité**.
Donné p premier, choisir $P(x) = x - a$ et vérifier l'équation.
Coûteux en temps dans le pire des cas !
Amélioration : évaluer la congruence modulo $x^r - 1$ pour un premier r bien choisi (dans l'anneau $\mathbb{F}_p[x]/x^r - 1$). Une itération de l'algorithme sera :

$$(x-a)^p \equiv (x^p - a) \mod (x^r - 1, p)$$

Et de vérifier cette nouvelle congruence pour un petit nombre de valeurs de a (de l'ordre de $O(\sqrt{r}\log p)$).

Preuve de l'identité

$$\text{si } \gcd(a,p) = 1, \qquad [(x-a)^p \equiv (x^p - a) \mod p \Leftrightarrow p \text{ premier}]$$

Si p premier, $p \mid \binom{p}{r} = \frac{p(p-1)!}{r!(p-r)!}$ pour $1 \leq r \leq p-1$ et $(x-a)^p \equiv (x^p - a^p) \mod p \equiv x^p - a \mod p$ par Fermat-Euler.
Réciproquement, supposons p composé. Alors il existe q premier diviseur de p et soit q^k la plus grande puissance de q qui divise p. q^k ne divise pas $\binom{p}{q} = \frac{p!}{q!(p-q)!} = \frac{q^k \alpha (p-1)!}{q(q-1)!(p-q)!}$ et q^k est premier avec a^{p-q}. Le coefficient de x^q dans le développement de $(x-a)^p$ n'est pas nul alors qu'il l'est dans le membre droit de l'identité, une contradiction.

Entrée : un entier $n > 1$
Si n est de la forme $a^b, b > 1$ **alors** n est COMPOSE **fsi**
$r \leftarrow 2$
Tantque $r < n$ **faire**
  Si $\gcd(n,r) \neq 1$ **alors** n est COMPOSE **fsi**
  Si r est premier ≥ 2 **alors**
    soit q le plus grand facteur premier de $r-1$;
    Si $(q \geq 4\sqrt{r}\log n)$ **et** $(n^{\frac{r-1}{q}} \neq 1 \mod r)$ **alors** ARRET ;
  fsi
  $r \leftarrow r + 1$
ftq
Pour $a \leftarrow 1$ **jusqu'à** $2\sqrt{r}\log n$ **faire**
  Si $(x-a)^n \not\equiv (x^n - a) \mod (x^r - 1, n)$ **alors** n est COMPOSE **fsi**
fpour
n est PREMIER

Exponentiation modulaire pour calculer $a^b \mod n$

Algorithme

Exponentiation modulaire (a, b, n)
$d \leftarrow 1$;
Soit $\langle b_k, b_{k-1}, \dots b_0 \rangle$ *la représentation binaire de* b
Pour $i \leftarrow 0$ **jusqu'à** k **faire**
  $d \leftarrow (d.d) \mod n$;
  si $b_i = 1$ **alors**
    $d \leftarrow (d.a) \mod n$;
renvoie d

```
1  def expMod(a,b,n):
2      d = 1
3      for i in bin(b)[2:] :
4          d = (d*d) % n
5          if i == '1':  d = (d*a) % n
6      return(d)
```


A la main

$17^{71} \mod 133$. $71 = \langle 1000111 \rangle$

i	b_i	17^{2^i}	17^{2^i}	mod 133	valeur
0	1	17	17	mod 133	17
1	1	17^2	289	mod 133	23
2	1	23^2	529	mod 133	130
3	0	130^2	16900	mod 133	9
4	0	9^2	81	mod 133	81
5	0	81^2	6561	mod 133	44
6	1	44^2	1936	mod 133	74

et $17^{71} = 17.17^2.17^{2^3}.17^{2^6} = 17.23.130.74 \mod 133 = 47$.

Chiffrement RSA

1. choisir p et q premiers assez grands -de l'ordre de 10^{100}
2. fixer $n = pq$ et publier n
3. calculer $\varphi(n) = (p-1)(q-1)$
4. choisir et publier $e / \gcd(e, \varphi(n)) = 1$ (clé publique, *encipher*)
5. calculer d tq $d.e \equiv 1 \mod \varphi(n)$ (clé privée, *decipher*)

Chiffrer : $E : M \mapsto M^e \mod n$.
Déchiffrer : $D : C \mapsto C^d \mod n$ (d est la trappe).
Implémentations : logicielles, matérielles et même mixtes.
Sur des cartes dédiées, RSA environ 1000 fois plus lent que DES.

Preuve de RSA

Avec $ed \equiv 1 \mod \varphi(n)$,
$M^{ed} \equiv M^{k\varphi(n)+1} \mod n$
Par Euler, $M^{\varphi(p)} \equiv 1 \mod p \Rightarrow M^{p-1} \equiv 1 \mod p$ si $\gcd(M, p) = 1$
Et comme $(p-1) | \varphi(n)$,
$M^{k\varphi(n)+1} \equiv M \mod p$
(trivialement vrai si $M \equiv 0 \mod p$)
De même pour q :
$M^{k\varphi(n)+1} \equiv M \mod q$
(trivialement vrai si $M \equiv 0 \mod q$)
On a : $M^{ed} \equiv M \mod (pq)$

Exemple

A et B utilisent RSA avec $n = 133$. La clé publique de A est 35.

1. quelle est sa clé privée ?
2. si le chiffré reçu par A est 17, retrouver le clair. (Les messages sont compris entre 0 et $n-1$).

Clé privée de A ? Factoriser n en produit de 2 premiers ; on trouve $n = 133 = 7 \times 19$. $\varphi(133) = 6.18 = 108$.
Les premiers nombres premiers sont $1, 3, 5, 7, 11, 13, 17, 19 \ldots$
Avec $e = 35$, $d \equiv e^{-1} \mod \varphi(n)$. Euclide $(35, 108)$ donne les coefs de Bezout $(-37, 12)$. D'où $d = -35 = 108 - 37 = 71 \mod 108$.
Cryptanalyse de $A = 17$, il suffit de calculer $17^{71} \mod 133 = 47$

Textbook RSA

- $\text{Gen}(1^\tau)$ construit p, q premiers tq $|p| = |q| \approx \frac{1}{2c^3}\tau^3$ et donne pk=(e, n) et sk=(p, q, n, e, d) (c petite constante ≈ 8 [1]).
- $\text{Enc}_{\text{pk}}(\text{m})$ pour $m \in \{0,1\}^\star$ découpe m en $|m|/(|n|-1)$ blocs de $|n|-1$ bits en on applique le chiffre à chaque m_i
- $\text{DEC}_{\text{sk}}(c)$ déchiffre bloc par bloc les c_i

Hypothèse (hypothèse RSA)

Calcul infaisable : pour un τ assez grand, le calcul de m sur l'entrée $((e, n), c)$ avec RSA défini comme précédemment et $m \xleftarrow{\mathcal{U}} \mathbb{Z}_n^\star$.

Correspond à la définition de OW. Retour def. OW

1. valeur de c fixée par algo. de factorisation

Attaque sur les paramètres

Cycles : Eve observe $c = m^e \mod n$; elle essaye de trouver ν t.q.

$$c^{e^\nu} \equiv c \mod n \Leftrightarrow e^\nu \equiv 1 \mod \varphi(n)$$

Ce qui permet de trouver $m \equiv c^{e^{\nu-1}} \mod n$
En effet $c^{e^\nu} \equiv c \mod n \Leftrightarrow c^{e^\nu - 1} \equiv 1 \mod n$ et, par d'Euler, on a $e^\nu - 1 \equiv 0 \mod \varphi(n) \Leftrightarrow e^\nu \equiv 1 \mod \varphi(n)$. Comme $c = m^e \mod n$ et $de \equiv 1 \mod \varphi(n)$, on peut prendre $d = e^{\nu-1}$.

Exemple

Publics : e et n, 35 et 133. Eve intercepte $c = 69$ et calcule

i	2	3	4
c^{e^i}	69	27	69

Il n'y a plus qu'à « lire » m pour $i = 3$, soit 27 (cycle de long. 2).

Low exponent attacks

On n'a pas toujours besoin de factoriser n

- dans le cas ou le clair m est chiffré par des PK $(e, n_1), (e, n_2), (e, n_3) \ldots$ avec un même e.

$$c_1 = m^e \mod n_1, \quad c_2 = m^e \mod n_2, \quad c_3 = m^e \mod n_3$$

- possible de retrouver m à partir des chiffrés
- des implémentations de RSA utilisent $e = 3$ (ou 65537 `ssl`)

Eve peut retrouver m ainsi en supposant les n_i premiers 2 à 2 (sinon Eve pourrait factoriser l'un des n_i). Par le Th. Chinois des restes, Eve calcule $c = m^3 \mod n_1 n_2 n_3$. Comme $m < n_1, n_2, n_3$, $m < n_1 n_2 n_3$ d'où $c = m^3$ qui devient une équation sur $\mathbb{N}$. m peut être retrouvé en calculant la racine cubique de c.
Correctif : utiliser du padding.

Correctif de sécurité RSA-OAEP

Utiliser du random padding comme OAEP (1995) permet de contrer les attaques "low exponent".
Optimal Asymmetric Encryption Padding est une forme de chiffre de Feistel qui utilise une source aléatoire et 2 fonctions de hachage (oracle aléatoire).
Pour un clair M, un aléa r et g, h deux fonctions de hachage :

$$\text{OAEP}(M) = (M \oplus g(r))||(r \oplus h(M \oplus g(r)))$$

Retrouver M connaissant, n la longueur de M :

- découper OAEP(M) en B_1 de longeur n et B_2
- $r = h(B_1) \oplus B_2$
- $M = B_1 \oplus g(r)$

Utilisé avec RSA, c'est RSA-OAEP, prouvé sémantiquement sûr.
Mais avec quelles fonctions de hachage ? (elles sont définies à partir de SHA1)

Si le module n est commun à tous les utilisateurs

On distribue un message commun à deux utilisateurs. m est envoyé aux deux utilisateurs de paramètres publics respectifs (e_1, n) et (e_2, n) qui vérifient $\gcd(e_1, e_2) = 1$.
Fred intercepte m^{e_1} et m^{e_2}. Il ne lui reste plus qu'à calculer avec Euclide étendu $u, v \in \mathbb{N}$ tq $ue_1 + ve_2 = 1$ le clair :

$$(m^{e_1})^u (m^{e_2})^v \equiv m \mod n$$

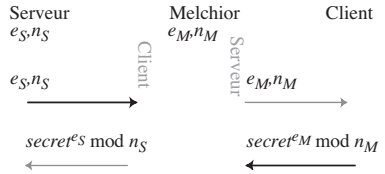
Exemple

le module partagé est 143 *et les exposants de chiffrement respectifs sont* $e_1 = 17$ *et* $e_2 = 19$. *On distribue* $m = 123$ *en envoyant* $123^{17} = 106$ *au premier utilisateur et* $123^{19} = 72$ *au second. Fred calcule alors les coefficients de Bezout du pgcd*$(17, 19) = (9, -8)$ *et retrouve* $m = 106^9 72^{-8} \mod 143 = 123$.

L'attaque de l'homme du milieu ("man in the middle")

Porte sur la communication des clés.

- Bob (client) demande ses paramètres publics à Alice (serveur)
- Alice envoie (e_S, n_S) à Bob
- Melchior intercepte (e_S, n_S) et envoie ses paramètres (e_M, n_M)
- Bob chiffre en utilisant à son insu (e_M, n_M) et envoie c
- Melchior intercepte le message c et le déchiffre en secret
- Melchior rechiffre secret avec (e_S, n_S) et le transmet à Alice...



Ah ! si Bob avait pu s'assurer que les données venaient d'Alice.

Attaque à $\varphi(n)$ connu

Connaître $(n, \varphi(n))$ revient à connaître la factorisation de n [4].
En effet, en posant : $\begin{cases} n = pq \\ \varphi(n) = (p-1)(q-1) \end{cases}$ et $q = \frac{n}{p}$:

$$\varphi(n) - (p-1)\left(\frac{n}{p} - 1\right) = 0 \Leftrightarrow p^2 + p\,(\varphi(n) - n - 1) + n = 0$$

équation du second degré de solutions p et q.
Calcul de $\varphi(n)$ aussi difficile que la factorisation de n.

Exemple

Connus $n = p.q = 133$ *et* $\varphi(n) = 108$. *On pose*
$\varphi(n) - (p-1)\left(\frac{n}{p} - 1\right) = 0 \Leftrightarrow p^2 + p\,(\varphi(n) - n - 1) + n = 0$
Soit $p^2 + p(108 - 133 - 1) + 133 = 0 \Leftrightarrow p^2 - 26.p + 133 = 0$ *de discriminant* $\Delta = (-26)^2 - (4.133) = 144 = 12^2$ *et de solutions* $p = \frac{26 \pm 12}{2} = \{19, 7\}$.

Sûreté calculatoire

RSA est aussi sûr que la factorisation de n est difficile.
Complexité de quelques ≪ bons ≫ algorithmes de factorisation :

crible quadratique	$O(e^{((1+o(1))\sqrt{\log n \log\log n})})$
courbes elliptiques	$O(e^{((1+o(1))\sqrt{2 \log p \log\log p})})$
crible algébrique	$O(e^{((1{,}92+o(1))(\log n)^{1/3}(\log\log n)^{2/3})})$

(p : plus petit facteur premier de n).

Insécurité prouvée de RSA

- RSA n'est pas sémantiquement sûr
- $\mathrm{Enc}_{pk}(m)$ révèle des informations sur m !
  - il suffit de connaitre le symbole de Jacobi de m [2] :

$$\left(\frac{m^e}{n}\right) = \left(\frac{m^e}{p}\right)\left(\frac{m^e}{q}\right) = \left(\frac{m}{p}\right)\left(\frac{m}{q}\right) = \left(\frac{m}{n}\right)$$

- vient du fait que le chiffrement par RSA est déterministe !
- pourquoi un chiffrement déterministe empèche-t'il la sécurité sémantique ?

Plus de détails en suivant le lien

2. permet de décider si un entier modulaire n'est pas un carré

Sécurité sémantique

Pour qu'un chiffre à clé publique soit sémantiquement sûr (IND-CPA), il faut que l'adversaire (avec une puissance calculatoire limitée) soit incapable d'obtenir des informations significatives sur le clair à partir de la donnée du du chiffré et de la clé publique. La sécurité sémantique ne considère que le cas d'un adversaire ≪ passif ≫ qui observe les chiffrés.

Distinguabilité

- contredire IND-CPA : Ciphertext Distinguishability Pb (CDP)
- chiffre sémantiquement sûr si CDP est infaisable
- (attaquant peut avoir à sa disposition un dictionnaire clair/chiffrés et choisir le défi parmi eux.)

Plan

Un autre problème difficile

DLP : problème du logarithme discret de y en base g

Problème

INSTANCE : $g, y \in G$, *groupe fini*
QUESTION : *trouver* x *tel que* $g^x \equiv y$ *dans* G

ou, pour p un grand premier, g un générateur de $G = \mathbb{Z}_p^\star$,

$$g^x \equiv y \mod p \text{ et } x = \log_g(y) \mod p$$

Plus généralement, tout $y \in G$ possède un logarithme discret en base g ssi G cyclique de générateur g.

Exemple

Soit $G = \mathbb{Z}_7^\star$ groupe cyclique, d'ordre 6.

- en base 2, seuls $1, 2$ et 4 possèdent un logarithme discret ;
- en base g=3, on obtient le tableau :

nombre y	1	2	3	4	5	6
logarithme	6	2	1	4	5	3

Par exemple, pour nombre $= 1$ et log $= 6$, cela signifie que $\log_3 1 = 6$ (dans G), ce qu'on vérifie par $3^6 \mod 7 = 1$.

Calcul du logarithme discret – Shanks

S'applique à tout groupe fini G.
Complexité en temps $O(\sqrt{|G|}\log|G|)$ en espace $O(\sqrt{|G|})$
Idée : construire deux listes de puissances de g :

- une liste de petits pas $\{g^i : i = 0..\lceil\sqrt{n}\rceil - 1\}$ avec $n = |G|$
- une liste de pas de géant $\left\{y\left(g^{-\lceil\sqrt{n}\rceil j}\right) : j = 0..\lceil\sqrt{n}\rceil\right\}$.

Puis trouver un terme commun aux 2 listes. Ainsi,

$$g^{i_0} = y(g^{-j_0\lceil\sqrt{n}\rceil}) \text{ et } m = i_0 + j_0\lceil\sqrt{n}\rceil$$

Calcul du log. discret de groupes de faible cardinalité facile, opération difficile quand le cardinal de G croît.

Exemple

On travaille dans $\mathbb{Z}_{113}^{\times} =< 3 >$ d'ordre $n = 112$; $\sqrt{n} = r = 11$. On cherche le logarithme discret de $y = 57$ en base $g = 3$:
Liste (non ordonnée) des petits pas, forme (exposant, valeur) :

$$B = \{(0,1),(\mathbf{1},\mathbf{3}),(2,9),(3,27),(4,81),\\ (5,17),(6,51),(7,40),(8,7),(9,21),(10,63)\}$$

Liste (non ordonnée) des pas de géant, forme (exposant, valeur) :

$$L = \{(0,57),(1,29),(2,100),(3,37),(4,112),(5,55),(6,26),\\ (7,39),(8,2),(\mathbf{9},\mathbf{3}),(10,61),(11,35)\}$$

3 valeur commune aux 2 listes, engendré pour $i_0 = 1$ dans B et $j_0 = 9$ dans L. Le logarithme discret est $x = i_0 + r.j_0 = 100$.
Vérification : on calcule $g^x \mod 113 = 57$.


```
from sympy import *
g,n,r,y=3,113,11,57
B=[(i,pow(g,i,n)) for i in range(r)]
L=[(j,y*gcdex(pow(g,j*r,n),n)[0]\%n) for j in range(r+1)]
B=sorted(B, key=lambda x:x[1])
L=sorted(L, key=lambda x:x[1])
print(B,L)
```

```
[(0, 1), (1, 3), (8, 7), (2, 9), (5, 17), (9, 21), (3, 27), (7, 40), (6, 51), (10, 63), (4, 81)]
[(8, 2), (9, 3), (6, 26), (1, 29), (11, 35), (3, 37), (7, 39), (5, 55), (0, 57), (10, 61), (2, 100), (4, 112)]
```

```
1+r*9
```

100

Shanks

Algorithme :
Entrée : n le cardinal de G,g et $y \in G$
Sortie : logarithme discret de y en base g dans G. $r := \lceil \sqrt{n} \rceil$
Construire la liste $B := \{g^i : i = 0..\lceil \sqrt{n} \rceil - 1\}$
Construire la liste $L := \{y(g^{-\lceil \sqrt{n} \rceil j}) : j = 0..\lceil \sqrt{n} \rceil\}$
Trier les listes B et L selon un ordre sur les g^i et les g^{-rj}.
Trouver i_0 et j_0 tel que : $g^{i_0} = yg^{-rj_0}$.
RETURN$(i_0 + j_0 r)$.
Preuve :
Pour $r = \lceil \sqrt{n} \rceil$, on a :
Construction des listes : $r + 1 + r$ opérations de groupes : $O(r)$
Tri des listes : $O(r \log(r))$
Recherche d'un même élément dans deux listes triées : $O(\log(r))$ $\square$

Plan

Le chiffre d'El Gamal [3]

Repose sur DLP.

1. choisir p premier t.q. DLP est difficile dans $\mathbb{Z}_p{}^\star$
2. choisir un générateur $\alpha \in \mathbb{Z}_p{}^\star$
3. choisir $2 \leq a < p-1$, la **clé privée**
4. calculer $\beta \equiv \alpha^a \mod p$
5. **clé publique :** p, α, β.

Chiffrer : $E : (x, k) \mapsto (y_1 = \alpha^k \mod p, y_2 = x\beta^k \mod p)$
pour k aléatoire secret de $\mathbb{Z}_{p-1}{}^\star$
Déchiffrer : $(y_1, y_2) \mapsto y_2 (y_1{}^a)^{-1} \mod p$

ElGamal Rockstar

Fonctionnement

Alice écrit à Bob en utilisant $pk = (p, \alpha, \beta)$

$$E : (x, k) \mapsto (y_1 = \alpha^k \mod p, y_2 = x\beta^k \mod p)$$

k aléatoire secret de $\mathbb{Z}_{p-1}{}^\star$ (on a donc un chiffrement probabiliste).
Clair x est « masqué » par β^k.
Valeur de α^k transmise comme partie du chiffré comme y_1.

Bob, avec sk a, calcule $\beta^k = \alpha^{ak}$. Comme $\beta = \alpha^a$, il calcule $(\alpha^k)^a = \beta^k$. Reste à multiplier y_2 par $(\beta^k)^{-1} \mod p = x$.

Observons que le chiffré est deux fois plus long que le clair.

Exemple

Soit $p = 2579, \alpha = 2$, $a = 765$. $\beta = 2^{765} \mod 2579 = 949$.
Alice veut transmettre $x = 1299$ à Bob.
Elle choisit $k \in \mathbb{Z}^\star_{p-1} = 853$ et calcule (mod 2579) :

$$y_1 = 2^{853} = 435 \qquad y_2 = 1299.(949)^{853} = 2396$$

Bob reçoit $(435, 2396)$ et connait $a = 765$. Il calcule $(y_1)^a$ mod $p = 435^{765} \mod 2579 = 2424$, cherche l'inverse mod p par Euclide étendu : $(2424, 2579) = -599.2424 + 563.2579 = 1 \Leftrightarrow (\beta^k)^{-1} = -599 = 1980$.
Puis il calcule $2396.1980 \mod 2579 = 1299$.

Difficulté du logarithme discret

$G = \langle g \rangle$ d'ordre $p-1$, $\forall y \in G, \exists! x : g^x = y$ (on note $x = \log_g y$)
DLP : donnés g, y, calculer x dans G.

$\mathcal{G}$ algo polytime : entrée 1^n retourne $G = \langle g \rangle$ d'ordre $p-1$.
Expérience $\mathrm{DLog}_{A,\mathcal{G}}(n)$:

- lance $\mathcal{G}(1^n)$ pour avoir (G, p, g) (générateur de groupe)
- $y \xleftarrow{u} G$
- A reçoit (G, p, g, y) et retourne x
- résultat expérience 1 si $g^x = y$ sinon 0

Définition (Hypothèse DH)

DLP est difficile pour $\mathcal{G}$ si pour tout algo A PPT, il existe negl : $Pr(DLog_{A,\mathcal{G}}(n) = 1) \leq negl(n)$

Les problèmes CDH et DDH permettent de construire de bons $\mathcal{G}$

Computational Diffie Hellman CDH

$\langle g \rangle = G \quad y_1, y_2 \in G \quad DH_g(y_1, y_2) \stackrel{\Delta}{=} g^{\log_g y_1 \log_g y_2}$
$y_1 = g^x \quad y_2 = g^z \Rightarrow DH_g(y_1, y_2) = g^{xz} = (y_1)^z = (y_2)^x$

CDH = calculer $DH_g(y_1, y_2)$ pour y_1, y_2 choisis au hasard

Si DLP relatif à $\mathcal{G}$ facile, CDH aussi
Si log discret difficile, CDH difficile ?

Decisional Diffie Hellman DDH

DDH revient à distinguer $DH_g(y_1, y_2)$ d'un élément aléatoire de G : pour $y_1, y_2 \stackrel{u}{\leftarrow} G$ et y' une solution, DDH revient à décider si $y' = DH_g(y_1, y_2)$ ou si $y' \stackrel{u}{\leftarrow} G$.

Définition

DDH est difficile relativement à $\mathcal{G}$ si, pour tout algo D PPT, il existe negl. tq

$|Pr(D(G,p,g,g^x,g^y,g^z)=1) - Pr(D(G,p,g,g^x,g^y,g^{xy})=1)| \leq negl(n)$

pour $\mathcal{G}(1^n)$ qui renvoie (G,p,g) et $x,y,z \stackrel{u}{\leftarrow} \mathbb{Z}_p$

Lemme utile

Le multiple d'un élément y tiré uniformément est unif. distribué.
Ou, y' ne contient pas d'information sur m.

Lemme

G groupe fini, $m \in G$ qcq. Les distrib. de probabilité relatives à :

- $y \stackrel{u}{\leftarrow} G$ *et* $y' := m \cdot y$
- $y' \stackrel{u}{\leftarrow} G$

sont identiques. Autrement dit, $\forall \hat{y} \in G, Pr(m \cdot y = \hat{y}) = 1/\sharp G$

$\hat{y} \in G$ qcq. Alors $Pr(m \cdot y = \hat{y}) = Pr(y = m^{-1} \cdot \hat{y})$.
Comme $y \stackrel{u}{\leftarrow} G$, la proba. pour que y soit un élément donné de G est $1/\sharp G$ □

ElGamal (rappel)

Le chiffre d'ElGamal Π est (mod p) :

- Gen $:1^n \mapsto \mathcal{G}(1^n) \mapsto (G,p,g), \quad a \stackrel{u}{\leftarrow} \mathbb{Z}_p^\star, \quad \beta := g^a$;
  $pk = (G,p,g,\beta)$ et $sk = (G,p,g,a)$
- E : reçoit (pk, m) ; $k \stackrel{u}{\leftarrow} \mathbb{Z}_p^\star$; renvoie $c = (y_1, y_2) = (g^k, m\beta^k)$
- D : reçoit sk et $c = (y_1, y_2)$; renvoie $m := y_2 (y_1^a)^{-1}$

Théorème

Si DDH est difficile pour $\mathcal{G}$, ElGamal est IND-CPA.

Comparer fonctionnement de Π à celui de Π' qui ressemble syntaxiquement à Π en remplaçant toutes ses sorties par des VA.

On montre IND-EAV plutôt que IND-CPA (équivalent).

IND-EAV : $\text{PubK}^{\text{EAV}}_{A,\Pi} \equiv \text{PubK}^{\text{CPA}}_{A,\Pi}$

1. $\text{Gen}(1^n)$ produit (pk, sk)
2. A reçoit pk et retourne $m_0, m_1 \in M(pk)$ de même long.
3. $b \xleftarrow{u} \{0,1\}$; $c \leftarrow E_{pk}(m_b)$ et envoyer c à A
4. A retourne un bit b'
5. A réussit l'expérience (i.e. renvoie 1) ssi $b = b'$

Définition

Π est CPA-sûr si, pour tout adversaire A PPT, il existe $negl(.)$ tq :

$$Pr(PubK^{CPA}_{A,\Pi}(n) = 1) \leq \frac{1}{2} + negl(n)$$

On pose : $\varepsilon(n) = Pr(\text{PubK}^{\text{EAV}}_{A,\Pi}(n) = 1)$ qu'il faut évaluer.

Le chiffre Π', pas déchiffrable mais convenable pour A

- Gen' =Gen rend $pk = (G, p, g, \beta)$ et $sk = (G, p, g, a)$
- E : donné (pk, m) ; $y, z \xleftarrow{u} \mathbb{Z}_p^\star$; rend $c = (y_1, y_2) = (g^y, mg^z)$

Par le lemme :

- y_2 unif. distribué sur G et indépendant de m
- y_1 est unif. distribué et indépendant de m.

On ne tire pas d'information sur m à partir de c :

$$Pr(\text{PubK}^{\text{EAV}}_{A,\Pi'}(n) = 1) = \frac{1}{2}$$

Algo D PPT qui résout DDH relativement à $\mathcal{G}$

D reçoit pk et c :$(G, p, g, g^x = \beta, g^y = y_1, g_3)$ avec

$g_3 = \begin{cases} g^{xy} \\ g^z \end{cases}$ où $x, y, z \xleftarrow{u} \mathbb{Z}_p^\star$.

- $pk := (G, p, g, g^x = \beta)$ et appelle A pour obtenir m_0, m_1
- $b \xleftarrow{u} \{0,1\}$; $y_1 := g^y$ et $y_2 := m_b g_3$
- donne (y_1, y_2) à A qui renvoie b'
- résultat expérience 1 si $b' = b$ sinon 0

Deux comportements possibles pour D selon g_3

$g_3 = g^z$ A, appelé par D fonctionnera ĉ ds $\text{PubK}^{\text{EAV}}_{A,\Pi'}(n)$ sur un chiffré de la forme (g^y, mg^z). Donc

$$Pr(D(\text{pk}, g^y, g^z) = 1) = Pr(\text{PubK}^{\text{EAV}}_{A,\Pi'}(n) = 1) = 1/2$$

$g_3 = g^{xy}$ A, appelé par D fonctionnera ĉ ds $\text{PubK}^{\text{EAV}}_{A,\Pi}(n)$ sur un chiffré de la forme $(g^y, m(g^x)^y)$. Donc

$$Pr(D(\text{pk}, g^y, g^{xy}) = 1) = Pr(\text{PubK}^{\text{EAV}}_{A,\Pi}(n) = 1) = \varepsilon(n)$$

par hyp. DDH difficile pour $\mathcal{G}$ donc $\exists$ negl. tq $\text{negl}(n) \geq$

$$\left| \begin{array}{l} Pr(D(\text{pk}, g^y, g^z) = 1) - \\ Pr(D(\text{pk}, g^y, g^{xy}) = 1) \end{array} \right| = |1/2 - \varepsilon(n)| \Rightarrow \varepsilon(n) \leq 1/2 + \text{negl}(n)$$

Partage des paramètres

Dans la définition d'ElGamal, on demande aux sujets de lancer Gen pour engendrer G, p, g. En pratique, ces paramètres sont souvent engendrés une fois pour toute.
P.e. un admin système peut fixer ces paramètres pour un paramètre de sécurité donné n et tout le monde peut partager ces valeurs.

Dans un BSD sous `/etc/moduli`. Demander `man moduli`

```
DESCRIPTION The /etc/moduli file contains prime
numbers and generators for use by sshd in the
Diffie-Hellman Group Exchange key exchange method.
```


G. Brassard.
Cryptologie contemporaine.
Logique, mathématiques, informatique. Masson, 1993.

W. Diffie and M.E. Hellman.
New directions in cryptography.
IEEE Trans. on Inform. Theory, 22(6) :644–654, 1976.

T. ElGamal.
A public-key cryptosystem and a signature scheme based on discrete logarihms.
IEEE trans. on Info. Theory, 31(4) :469–472, 1985.

N. Koblitz.
A course in number theory and cryptography.
Graduate texts in mathematics. Springer Verlag, 1987.

A. Salomaa.
Public Key Cryptography.
EATCS monographs. Springer Verlag, 1990.

G. Zémor.
Cours de cryptographie.
Cassini, 2000.