D31: Protocoles Cryptographiques

Le cryptosystème RSA

Nicolas Méloni

Master 2: 1er semestre
(2014/2015)

Un peu de théorie des nombres

Rappels

- $\mathbb{Z}/n\mathbb{Z} = \mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$
- $x \in \mathbb{Z}_n$ est inversible s'il existe y tel que:

$$x \times y \equiv 1 \mod n$$

- $x \in \mathbb{Z}_n$ est inversible si et seulement si $\gcd(x, n) = 1$
- On note $\mathbb{Z}_n^*$ le groupe multiplicatif des inversibles de $\mathbb{Z}_n$
- Si p est premier, alors $\mathbb{Z}_p$ est un corps et $\mathbb{Z}_p^*$ est cyclique
- On $\phi(n)$ (fonction d'Euler) le nombre d'éléments inversibles de $\mathbb{Z}_n$ (i.e. le cardinal de $\mathbb{Z}_n^*$)

Propriétés de la fonction ϕ

- Si n et m sont premiers entre eux alors $\phi(n \times m) = \phi(n) \times \phi(m)$
- Si p est un nombre premier, alors $\phi(p) = p - 1$ et $\phi(p^k) = p^{k-1}(p-1)$
- Soit $n = p_1^{\alpha_1} \dots p_r^{\alpha_r}$ la décomposition en facteurs premiers de n. Alors

$$\phi(n) = \prod_{p|n} (1 - \frac{1}{p})$$

Un peu de théorie des nombres

Quelques résultats

Théorème de Lagrange

Soient G un groupe multiplicatif d'ordre n et $g \in G$. Alors l'ordre de g divise n.

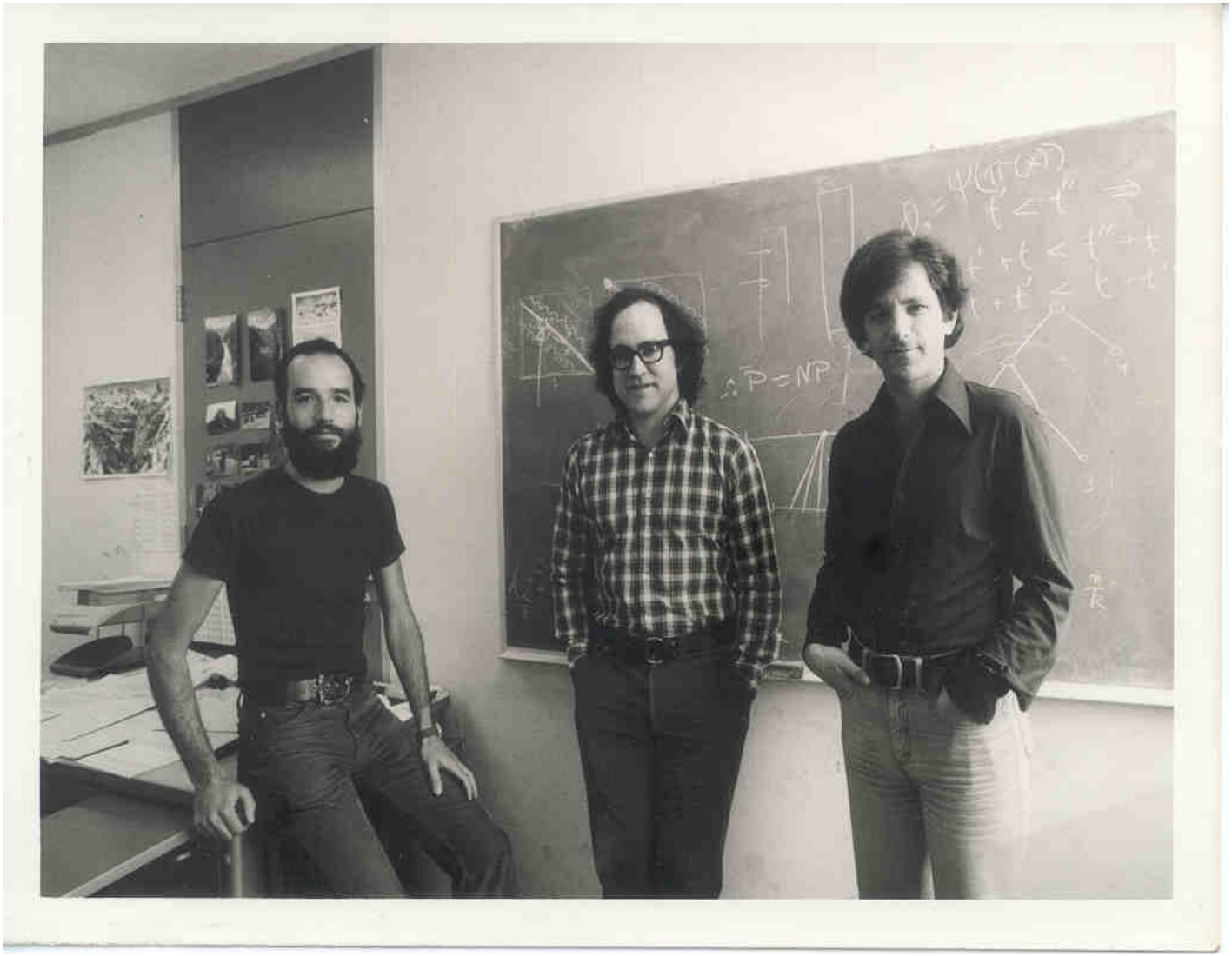
Corollaire 1

Si $b \in \mathbb{Z}_n^*$, alors $b^{\phi(n)} \equiv 1 \mod n$.

Corollaire 2

Soient p un nombre premier et $b \in \mathbb{Z}_p$. Alors $b^p \equiv b \mod p$.

RSA (Rivest - Shamir - Adleman)



RSA (Rivest - Shamir - Adleman)

Le cryptosystème

Soit $n = pq$, où p et q sont deux nombres premiers. Soit $\mathcal{M} = \mathcal{C} = \mathbb{Z}_n$. Soient (a, b) un couple d'entiers tels que $ab \equiv 1 \mod \phi(n)$. On définit les données:

$$K_{pub} = (n, b), K_{sec} = (p, q, a)$$

Et les algorithmes de chiffrement/déchiffrement:

$$E(K_{pub}, m) = m^b \mod n$$

$$D(K_{sec}, c) = c^a \mod n$$

RSA (Rivest - Shamir - Adleman)

Génération des paramètres

1. Générer deux grands nombres premiers, p et q, tels que $p \neq q$
2. $n \leftarrow pq$ et $\phi(n) \leftarrow (p-1)(q-1)$
3. Générer aléatoirement un entier b premier avec $\phi(n)$
4. $a \leftarrow b^{-1} \mod \phi(n)$
5. $K_{pub} = (n, b)$ et $K_{sec} = (p, q, a)$

Sécurité du système RSA

La sécurité repose sur la difficulté d'inverser la fonction:

$$e_b : x \mapsto x^b \mod n.$$

- Factoriser n permet d'obtenir facilement $\phi(n)$ et donc a
- Il est conjecturé que le problème de l'inversion de e_b est équivalent à la factorisation de n
- Aucune preuve à ce jour

Question

A quel point est-il difficile de factoriser ?

Factorisation d'entiers

Algorithme naïf

Diviser l'entier n par tous les nombre premiers inférieurs à $\sqrt{n}$.
Compléxité $O(\sqrt{n})$.

- Rapidement inutilisable pour de grands entiers
- Aucun algorithme polynomial, mais des algorithmes de compléxité intermédiaire existent

Compléxité sous-exponentielle

$$L_n[\alpha, c] = O\left(e^{(c+O(1))(\ln n)^{\alpha}(\ln \ln n)^{(1-\alpha)}}\right)$$

Algorithmes de factorisation

Crible quadratique

Compléxité: $L_n[\frac{1}{2}, 1]$

Courbes elliptiques

Compléxité: $L_p[\frac{1}{2}, \sqrt{2}]$, où p est le plus petit facteur premier de n

Crible algébrique

Compléxité: $L_n[\frac{1}{3}, 1.92]$

Un exemple simple

L'algorithme $p-1$ de Pollard

Permet la recherche de facteurs de n plus petit qu'une certaine borne B.

On considère p un diviseur de n tel que $p-1 = p_1^{\alpha_1} \dots p_r^{\alpha_r}$ avec $p_i^{\alpha_i} \leq B$.

- On a $p-1 \mid B!$,
- on calcule $a \equiv 2^{B!} \mod n$,
- on remarque que $a \equiv 1 \mod p$,
- donc $p \mid a-1$ et $p \mid n$ et donc $p \mid \gcd(a-1, n)$.

Un exemple simple

Algorithm 1 Algorithme $p-1$ de Pollard

Require: Une borne B et entier n

1: $a \leftarrow 2$
2: **for** $j = 2 \dots B$ **do**
3: $\quad a \leftarrow a^j \mod n$
4: **end for**
5: $d \leftarrow \gcd(a-1, n)$
6: **if** $1 < d < n$ **then**
7: $\quad$ **return** d
8: **else**
9: $\quad$ **return** Echec
10: **end if**

Un exemple simple

Analyse de la compléxité

- Effectue B exponentiations modulaires
- Chaque exponeniation a une compléxité de $O(\ln(n)^3)$
- Méthode efficace tant que B est petit i.e. de l'ordre $O(\ln(n)^i)$

- Pour contrer cette méthode, il suffit de choisir $p = 2p_0 + 1$ et $2q_0 + 1$ avec p_0 et q_0 premiers.

Sécurité sémantique

- La fonction de chiffrement est déterministe et n'est donc pas sure sémantiquement.
- Certaines informations fuitent, par exemple si $y \equiv x^b \mod n$ alors

$$\left(\frac{y}{n}\right) = \left(\frac{x^b}{n}\right) = \left(\frac{x}{n}\right)$$

où $\left(\frac{x}{n}\right)$ est le symbole de Jacobi.

Attaque à chiffré choisi

Soit $c \equiv m^b \mod n$ un chiffré.

- L'attaquant choisi un nombre $r \in \mathbb{Z}_n^*$,
- il demande le déchiffrement de $c' \equiv r^b \times c \mod n$,
- il reçoit alors

$$m' \equiv (c')^a \equiv r^{ab} \times c^a \equiv r \times m \mod n$$

- il en déduit

$$m \equiv m' \times r^{-1} \mod n$$

.

Quelques failles classiques de RSA

Module commun

On suppose que Alice et Bob partagent le même module RSA n. On note (e_A, d_A) (resp. (e_B, d_B)) le couple clé publique / clé privée d'Alice (resp. de Bob).

Module commun 1: déchiffrement d'un chiffré

On suppose que $\gcd(e_A, e_B) = 1$. On suppose également qu'un même message x est envoyé à Alice et Bob. On a deux chiffrés $y_A \equiv x^{e_A} \mod n$ et $y_B \equiv x^{e_B} \mod n$. Un attaquant peut alors déchiffrer le message, en effet:

- d'après le théorème de Bezout; $\exists (u, v)$ tq $ue_A + ve_B = 1$.
- On a alors

$$\begin{aligned} (y_A)^u (y_B)^v &\equiv x^{ue_A} \times x^{ve_B} \mod n \\ &\equiv x^{ue_A + ve_B} \mod n \\ &\equiv x \mod n \end{aligned}$$

Module commun 2: calcul de la clé de chiffrement

Bob peut calculer la clé secrête de d'Alice à partir de n, e_B, d_B et e_A.

Soit k tel que $e_B d_B - 1 = k\phi(n)$. Soit t tel que $t \mid e_B d_B - 1$ et $\gcd\left(\frac{e_B d_B - 1}{t}, e_A\right) = 1$. Alors

$$\exists (u, v) : u\left(\frac{e_B d_B - 1}{t}\right) + v e_A = 1.$$

En choisissant $v > 0$ on obtient

$$v e_A \equiv 1 \mod \phi(n).$$

Algorithm 2 Algorithme de calcul de t

Require: n, e_B, d_B et e_A

$g \leftarrow e_B d_B - 1$
$h \leftarrow \gcd(g, e_A)$
$t \leftarrow 1$
while $h \neq 1$ **do**
    $g \leftarrow g/h$
    $h \leftarrow \gcd(g, e_A)$
    $t \leftarrow th$
end while
return t

Petits exposants publics

On suppose que A_1, A_2 et A_3 possèdent des modules RSA distincts n_1, n_2 et n_3 mais la même clé de chiffrement $e = 3$. On suppose qu'ils reçoivent y_1, y_2 et y_3, trois chiffrés du même message x. Ainsi

$$\forall i \in \{1, 2, 3\}, y_i \equiv x^3 \mod n_i$$

En supposant les n_i premiers deux à deux le théorème des restes chinois permet de calculer $y \equiv x^3 \mod n_1 n_2 n_3$. Or comme $\forall i, x < n_i$, on a $x^3 < n_1 n_2 n_3$ et donc $y = x^3$. Un calcul de racine cubique dans $\mathbb{R}$ permet d'obtenir x.

Quelques failles classiques de RSA

Petite clé privée: Attaque de Wiener

Attaque faisant appel aux fractions continues. Elle permet de calculer la clé de déchiffrement d si

$$d < \frac{1}{3} n^{\frac{1}{4}}.$$

Quelques failles classiques de RSA

Mauvaise génération de paramètres

En 2013 un groupe de chercheur à pu factoriser 184 clé RSA certifiées. La faute à un générateur aléatoire déficient. Il suffit pour cela de récuperer autant de clés publiques que possible puis d'effectuer un calcul de pgcd entre les différents clés.

Version sémantiquement sure de RSA

- (n, p, k, e, d): des paramètres RSA
- (E, D): fonction de chiffrement symétrique authentifié sur un espace $(\mathcal{K}, \mathcal{M}, \mathcal{C})$
- $H : \mathbb{Z}_n \to \mathcal{K}$: fonction de hachage parfaite (oracle aléatoire)

Version sémantiquement sure de RSA