Cryptosystèmes à clé publique

Plan

- Fonction à sens unique
- Brèche secrète
- Cryptosystème RSA

Fonction à sens unique

- Définition : fonction facile à calculer, mais difficile à inverser
  - $y=f(x)$ facile à calculer
  - $x=f^{-1}(y)$ difficile à calculer
- Exemple :
  - Les Morceaux = Casser(Assiette) : facile
  - Assiette = Recoller(LesMorceaux) : très difficile
- Autre exemple :
  - $y = x^2$: facile
  - $x = \sqrt{y}$: difficile

Fonction à sens unique

- En termes informatiques :
  - « facile » : calculable en temps polynomial
  - « difficile » : ne peut être calculé en temps polynomial
- De plus, si l'on connaît y et si l'on choisit x au hasard, la probabilité que y=f(x) doit être négligeable

Brèche secrète

- Fonction à sens unique à brèche secrète, ou fonction trappe
  - f(x) est facile à calculer
  - $f^{-1}(x)$ est très difficile à calculer, sauf si l'on connaît un élément
- Exemple :
  - Démonter un mécanisme complexe : facile
  - Remonter le mécanisme :
    - Facile si l'on dispose du plan de montage
    - Difficile sinon

RSA

- Peu de fonctions mathématiques peuvent être considérées comme des fonctions à sens unique
- Les nombres premiers apportent des solutions
- Leur répartition n'a pas encore d'explication
  - On sait estimer le nombre de nombres premiers :

$$\sim \frac{N}{\ln(N)}$$

  - Connaissant un nombre premier, on ne sait pas (efficacement) en déduire le suivant

RSA

- On sait facilement multiplier deux nombres entiers (même très grands)
- Connaissant un nombre entier (très grand), il est très difficile de le factoriser sous forme d'un produit de nombre premiers

Suite

- Principe : soit trois entiers n, e et d
  - Chiffrement grâce à la clé publique (e,n) :

$$E_K: C = P^e \ mod \ n$$

  - Déchiffrement grâce à la clé privée (d,n) :

$$D_K: P = C^d \ mod \ n$$

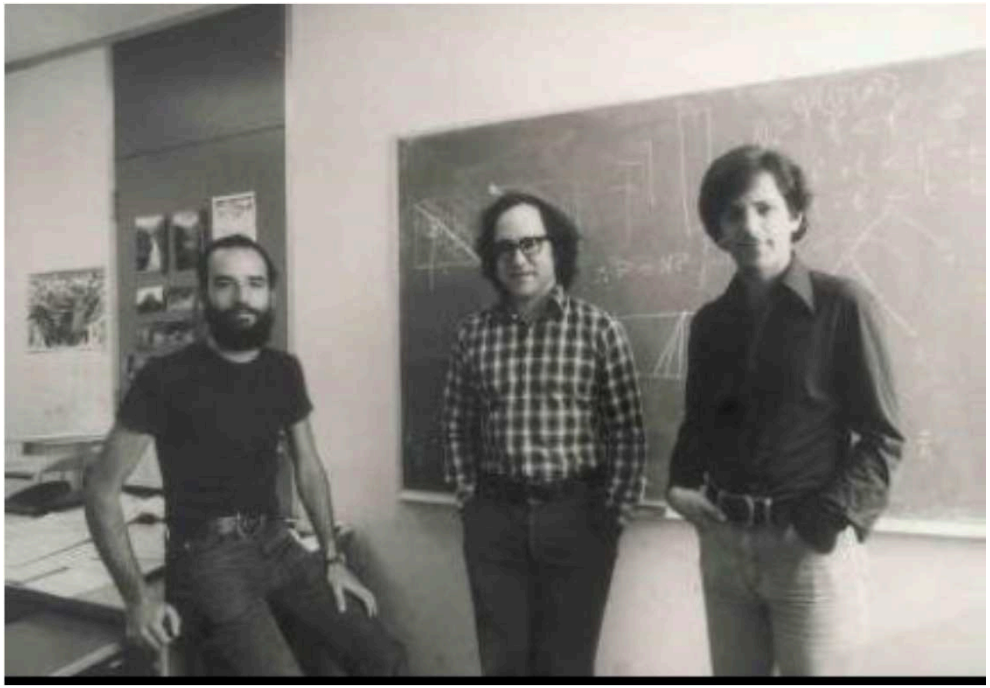
$$D_K: P = (P^d)^e \ mod \ n$$

$$D_K: P = P^{de} \ mod \ n$$

  - Comment construire e,d et n pour que $\mathbf{P^{ed} \ mod \ n}$ soit égal à **P** !!!

Suite

- Rivest, Shamir et Adleman (1977)
  - Ont imaginé ce système
  - Ont trouvé le moyen de construire e,d et n pour que E_K et D_K soient sûres



Suite

- Soit p et q deux nombres premiers
- On calcule $n = pq$
  - L'indicatrice d'Euler donne $\phi(n) = (p-1)(q-1)$
- On choisit e premier avec $\phi(n)$
  - Donc e est inversible mod $\phi(n)$
- On calcule d = $e^{-1}\ mod\ \phi(n)$
  - Donc ed = 1 mod $\phi(n)$

Suite

- On vérifie que $P^{ed}\ mod\ n = P$

$\text{ed} = 1 \bmod \phi(n)$

Donc $\exists k \in \mathbb{N} \quad ed = 1 + \phi(n) * k$

C'est-à-dire $ed = 1 + (p-1)(q-1) * k$

Donc $P^{ed}\ mod\ n = P^{1+(p-1)(q-1)*k}\ mod\ n$

Ou encore

$$P^{ed}\ mod\ n = P * (P^{(p-1)(q-1)})^{k}\ mod\ n$$

- On connaît le petit théorème de Fermat :
  - Si p est premier, et si a est premier avec p

    Alors $a^{p-1} \equiv 1 \, mod \, p$
- Ce théorème a été généralisé par Euler :
  - Si a est premier avec $n \in \mathbb{N}$

    Alors $a^{\phi(n)} \equiv 1 \, mod \, p$

- Or, dans RSA :

$$\phi(n) = (p-1)(q-1)$$

Donc $P^{ed} \bmod n = P * (P^{\phi(n)})^k \bmod n$

$$P^{ed} \bmod n = P * (1)^k \bmod n$$

Soit $P^{ed} \bmod n = P * (P^{\phi(n)})^k \bmod n$ par le théorème d'Euler

Finalement $P^{ed} \bmod n = P \bmod n$

- En résumé, e et d ne sont pas reliés par n (qui est connu) mais par $\phi(n)$ (qui est secret et qui dépend de p et q)
- Pour retrouver d à partir de e, il faut calculer son inverse par rapport à $\phi(n)$
- Pour obtenir $\phi(n)$, il faut pouvoir factoriser n en fonction de p et q (ce qui est très difficile pour des grands nombres

- RSA recommande des clés de 1024 bits à 2048 bits (soit des nombres décimaux ayant entre 340 et 680 chiffres)
- Pour implémenter RSA il faut :
  - Trouver des nombres premiers très grands
  - Pouvoir calculer l'exponentiation modulaire très efficacement avec un très grand exposant
  - Calculer l'inverse d'un nombre
    - On sait faire grâce à l'algorithme d'Euclide étendu, qui marche aussi avec les grands nombres

- Algorithme d'exponentiation modulaire
  - On souhaite calculer $c \equiv b^e \ mod \ n$
  - Algorithme intuitif : on calcule e multiplications puis on prend le modulo m
    - Complexité : O(e)
    - Problème : 5^{19} mod 9 = 5
    - 5^{19} = 1907348632812 5 n'est pas représentable avec un int, or 5^{19} mod 9 n'a qu'un seul chiffre !

- Algorithme d'exponentiation modulaire efficace
  - On considère la représentation binaire de l'exposant

$$e = \sum_{i=0}^{n-1} a_i 2^i$$

$$\text{Donc } b^e = b^{\left(\sum_{i=0}^{n-1} a_i 2^i\right)} = \prod_{i=0}^{n-1} \left(b^{2^i}\right)^{a_i}$$

$$\text{Finalement } c \equiv \prod_{i=0}^{n-1} \left[\left(b^{2^i}\right)^{a_i} \bmod m\right]$$

- Par convention, $a_{n-1} = 1$
- Si $a_i = 0$ alors $(b^{2^i})^0 = 1$ (on ne fait rien)
- On sait également que $b^{2^i} = (b^{2^{i-1}})^2 = b^{2^{i-1}} * b^{2^{i-1}}$
  - Pour calculer les b^{2^i} il suffira de faire une multiplication modulo m à chaque itération
- Complexité en O(log e)

- D'où l'algorithme efficace :

```
BigInt Outils::ExponentiationModulaire( BigInt b, BigInt e,
                                        const BigInt & m)
{
    BigInt  result = 1;

    while (e > 0)
    {
        if ((e % 2) > 0)
            result = (result*b) % m;
        e /= 2;
        b = (b*b) % m;
    }

    return result;
}
```

calcul de a^e mod n

- Basé sur la remarque suivante :
  - si e est pair, $a^e = (a^{e/2})^2$
  - si e est impair $a^e = (a^{e/2})^2 \times a$
- Algorithme d'exponentiation rapide modulaire
  1. Décomposer e en binaire : $e = \sum_{i=0}^{k} e_i 2^i$
  2. Calcul de $\{a^{2^i} \mod n\}_{0 \leq i \leq k}$
     - Utiliser la relation : $a^{2^{i+1}} = (a^{2^i})^2 \mod n$
  3. En déduire $a^e = \Pi_{i=0}^{k} (a^{2^i})^{e_i}$

Calcul de $51447^{21} \bmod 17$

$51447 = 3026 \times 17 + 5$ donc $(E) \equiv 5^{21} \mod 17$

1. Décomposition de 21 en binaire : $21 = 2^4 + 2^2 + 2^0$
2. Calcul de $\{5^{2^i} \mod 17\}_{0 \leq i \leq 4}$
   - $i = 0 : 5^{2^0} = 5 \mod 17$
   - $i = 1 : 5^{2^1} = 5^2 = 25 = 8 \mod 17$
   - $i = 2 : 5^{2^2} = 8^2 = 64 = 13 = -4 \mod 17$
   - $i = 3 : 5^{2^3} = (-4)^2 = 16 = -1 \mod 17$
   - $i = 4 : 5^{2^4} = (-1)^2 = 1 \mod 17$
3. On en déduit : $5^{21} = 5^{2^4} \times 5^{2^2} \times 5^{2^0} = 1 \times (-4) \times 5 = -20 = 14 \mod 17$

- Génération de clés
  - p = 11 q = 3
  - n = p * q = 33
  - $\varphi(n) = (p - 1)(q - 1) = 10 * 2 = 20$
  - K_c = 3 premier avec $\varphi(n)$
  - $K_d * K_c \equiv 1 \bmod 20$
  - → $K_d \equiv K_c^{-1} \bmod 20 = 7$
  - K_c publique et K_d secrète

- Chiffrement : M = 9
  - $C = M^{K_c} \bmod n = 9^3 \bmod 33$
    $= 15$
- Déchiffrement
  - C = 15
  - $M = C^{K_d} \bmod n = 15^7 \bmod 33$
    $= 9$

- Algorithme Euclide étendu(q,r) avec q<r
  - Q = (1,0)
  - R = (0,1)
  - Tant que r # 0 faire
    - t = q mod r
    - T = Q – [q/r] R
    - (q,r) = (r,t)
    - (Q,R) = (R,T)
  - ftq
  - Retourne (q,Q)
  - Fin
- Où la valeur q retournée donne la valeur du pgcd(q,r) et Q les coefficients de Bezout.

Calcul de $37^{-1}\ mod\ 108$

q	r	t=q mod r	Q	[q/r]	R	T
37	108	37	(1,0)	0	(0,1)	(1,0)
108	37	34	(0,1)	2	(1,0)	(-2,1)
37	34	3	(1,0)	1	(-2,1)	(3,-1)
34	3	1	(-2,1)	11	(3,-1)	(-35,12)
3	1	0	(3,-1)	3	(-35,12)	*
1	0	*	(-35,12)	*	*	*