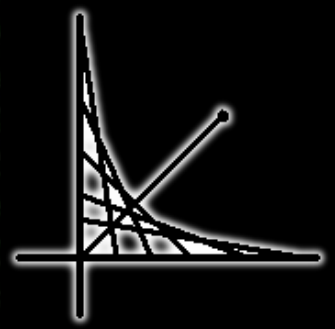
Cryptographie asymétrique

L'exemple de RSA

Trance
trancefusion@hotmail.fr
www.ghostsinthestack.org



Sommaire

- Introduction à la cryptographie
- Cryptographie symétrique
  - Principe, exemples, avantage / inconvénient
- Cryptographie asymétrique
  - Principe, illustration, utilisation courante
- L'exemple de RSA
  - Préliminaires, Principe
  - Attaques
- Algorithmes et nombres premiers

Cryptographie - Introduction

- But : échanger un secret entre 2 pers.
- Utilisation d'algorithmes simples
  - Algo révélé = perte du secret

  ➔ insuffisant
- Utilisation d'algo + d'une clé
  - Clé = sorte de « variante » pour l'algo
  - Algo révélé = nécessite toujours la clé...

  ➔ déjà mieux...

Cryptographie - Introduction

- Deux types de cryptographies
  - Crypto symétrique (à clé privée)
  - Crypto asymétrique (à clé publique)
- Chacune a ses avantages / inconvénients
- De nos jours, on utilise les 2

Cryptographie symétrique

- Nécessite :
  - Un algorithme F de chiffrement symétrique
    - Ou 2 algos (F1 pour chiffrer, F2 pour déchiffrer)
  - Une clé K partagée entre Bob et Alice
- Principe :
  - Bob chiffre le message M en C = F1(K,M)
  - Bob envoie le message C à Alice
  - Alice le décrypte et retrouve M = F2(K,C)

Cryptographie symétrique

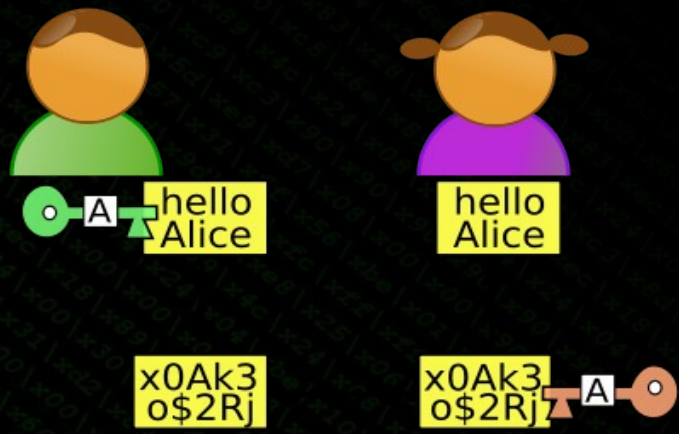
- Exemples simples
  - Code César et variantes (ROT13, ...)
    - F1(3,« COUCOU ») = « FRXFRX »
  - Algos de substitutions
  - Carré de Polybe...
- Avantage
  - Très rapide en général
- Inconvénient
  - Si la clé est interceptée, le secret est perdu

Cryptographie asymétrique

- Principe : On utilise 2 clés
  - Une clé « publique » qui sert à crypter
  - Une clé « privée » servant à décrypter
- Clé publique : peut transiter
  - S'assurer toutefois de son authenticité
- Clé privée : reste chez son propriétaire
- Il doit être impossible de déduire la clé privée de la clé publique

Principe en images

Alice génère deux clés.
La clé publique (verte) qu'elle envoie à Bob
et la clé privée (rouge) qu'elle conserve
précieusement sans la divulguer à quiconque.



Bob chiffre le message avec la clé publique d'Alice
et envoie le texte chiffré.
Alice déchiffre le message grâce à sa clé privée.

Cryptographie asymétrique

- Avantage
  - Même en interceptant le message, impossible de le décrypter sans la clé privée
- Inconvénient
  - Algos lents en général
  - Attaques par substitution de clé possibles...
    - Organismes de confiance, PKI, ...

Utilisation courante

- On utilise en fait les 2 types de crypto
- Crypto symétrique
  - On chiffre un message M avec une clé K
- Crypto asymétrique
  - On génère 2 clés privée (Pr) et publique (Pu)
  - La clé K est cryptée avec la clé Pu
  - On envoie M et la clé K cryptée sur le réseau
  - Le dest. décrypte la clé K avec Pr, et s'en sert pour finalement décrypter M
  - Gain de temps !

L'exemple de RSA

- Rivest, Shamir et Adleman en 1977
- Algo le plus utilisé
  - SSL
  - PGP
  - Carte Bleue
- Reste non cassé à ce jour
  - Attention à la signification de « casser »...

RSA - Préliminaires

- Diviseur
  - a est diviseur de b si b est un multiple de a
  - ex: 2 = diviseur de tous les nombres pairs
  - décomposition d'un nombre en ses diviseurs
- Nombre premier
  - n'a que 2 diviseurs : 1 et lui-même
  - ex: 2, 3, 5, 7, 11, 13, 17, 23, ...
- PGCD(a,b)
  - le plus grand diviseur commun de 2 a et b

RSA - Préliminaires

- Nombres premiers entre eux
  - a et b sont premiers entre eux ssi PGCD(a,b)=1
  - ex: 4 et 5, 3 et 10, ...
- Identité de Bézout
  - si a et b sont premiers entre eux, il existe x et y tel que a.x + b.y = 1

RSA - Préliminaires

- Congruences
  - On dit que « a est congru à b modulo n » , et l'on note $a \equiv b [n]$ si a peut s'écrire :

    $a = n.q + b \quad avec \quad 0 < b < n$
  - r est le reste de la division de a par n
  - q est le quotient
  - ex: 15 = 7*2 + 1 donc $15 \equiv 1 [7]$
  - Si $a \equiv 0 [n]$ alors a est un multiple de n

RSA - Préliminaires

- Fonction indicatrice d'Euler : Phi
  - A valeur de $\mathbb{N}$ dans $\mathbb{N}$ (entiers naturels)
  - Phi(n) = nombre d'entiers premiers à n et inférieurs à n
  - ex: Phi(8) = 3 car 3,5,7 premiers avec 8
  - Calcul de Phi(n) : Si **n = p.q** avec p et q premiers, alors **Phi(n) = (p-1).(q-1)**
  - Très dur à calculer si n est grand...
    - Calculer Phi(n) est aussi dur que de factoriser n

RSA - Préliminaires

- Théorème d'Euler
  - Si a est premier avec n, alors : $a^{Phi(n)} \equiv 1 [n]$
- Petit théorème de Fermat
  - Si p est un nombre premier, $a^{p} \equiv a [p]$

RSA – Génération des clés

- Choisir 2 entiers premiers n et p
- Poser N = p.q
- Calculer Phi(N) = (p-1).(q-1)
- Choisir E tq E<N et E premier avec Phi(N)
- Calculer D en résolvant $E.D \equiv 1 [Phi(N)]$
  - cf « Algo d'Euclide étendu » sur Wikipédia...
- C'est fini !
  - Le couple (N,E) est la clé publique
  - Le couple (N,D) est la clé privée

RSA - (Dé)cryptage

- Pour crypter un message M :
  - $C \equiv M^E [N]$
- Pour décrypter un message crypté C :
  - $M \equiv C^D [N]$
- Comment ça marche ?
  - Posons $C \equiv M^E [N]$ et $M \equiv C^D [N]$. On a alors :

$$C^D \equiv (M^E)^D [N] \equiv M^{E.D} [N]$$

Or, $E.D \equiv 1 [Phi(N)]$ donc E.D = k.Phi(N) + 1

Donc $C^D \equiv M^{1+k.Phi(n)} [N] \equiv M.(M^{Phi(n)})^k [N] \equiv M[N]$

(Euler + Fermat) $\equiv 1 [N]$

(Voir http://fr.wikipedia.org/wiki/Rivest_Shamir_Adleman pour la preuve détaillée)

Attaques de RSA

- Factorisation du module (N = p.q)
  - Si on parvient à retrouver p et q, on peut tout retrouver...
  - ... mais la factorisation d'un nombre est un problème complexe (complexité NP)
    - Le temps croît exponentiellement avec la taille de N
  - Plusieurs méthodes existent pour accélérer...
    - Crible algébrique (NFS)
  - ... mais restent encore trop lentes

Attaques de RSA

- Attaques temporelles (timing attacks)
  - Attaque uniquement liée à l'implémentation !
  - Tiennent compte du temps mis pour décrypter
  - Inefficace si l'implémentation de RSA utilise de l'aveuglement cryptographique (blinding)
    - Faire en sorte que le temps de calcul soit constant
    - Insérer des calculs « inutiles » dans l'algo...

Attaques de RSA

- Ordinateur quantique... ?
  - Nouvelle génération de calculateurs
  - Régis par la mécanique quantique
  - Utilisent des q-bits à états superposables
  - Accélération phénoménale de la vitesse de factorisation (temps linéaire)
  - Le record actuel : 15 = 5 * 3 :)
  - ... c'est pas encore gagné !

RSA, en résumé

- Un système basé sur une conjecture
  - On « suppose » qu'il est fiable, et que le casser revient à factoriser le module
- En pratique, a fait ses preuves
  - même si certaines implémentations contiennent des failles...
- Utiliser des clés de 1024 bits min.
- Lent, comme tout système asymétrique
  - Sert en pratique à crypter des clés symétriques pour les échanger

Algos et nombres premiers

- Buts
  - Générer des nombres premiers
    - En général de plus de 500 chiffres
  - Tester si un nombre et premier
  - Factoriser un nombre composé
- (Mal)heureusement...
  - Aucun algo 100% efficace
    - Les algos exacts sont trops lents
    - Les algos rapides ne sont pas toujours exacts

Générer des nombres premiers

- Il existe un tas de formules
  - toutes plus complexes les unes que les autres
- Ex : polynôme de degré 25 à 26 variables
  - Trouver la bonne combinaison des 26 variables qui donne un nombre positif
  - En réalité, inutile...
    - ... on n'a trouvé que le nombre 2 avec !

La bête...

$$
\begin{aligned}
&(1 \\
&- [\ w.z + h + j - q\ ]^2 \\
&- [\ 2.n + p + q + z - e\ ]^2 \\
&- [\ a^2.y^2 - y^2 + 1 - x^2\ ]^2 \\
&- [\ e^3.(e+2).(a+1)^2 + 1 - o^2\ ]^2 \\
&- [\ 16.(k+1)^3.(k+2).(n+1)^2 + 1 - f^2\ ]^2 \\
&- [\ ((a + u^2.(u^2 - a))^2 - 1).(n + 4.d.y)^2 + 1 - (x + c.u)^2\ ]^2 \\
&- [\ a.i + k + 1 - l - i\ ]^2 \\
&- [\ (g.k + 2.g + k + 1).(h + j) + h - z\ ]^2 \\
&- [\ 16.r^2.y^4.(a^2 - 1) + 1 - u^2\ ]^2 \\
&- [\ p - m + l.(a - n - 1) + b.(2.a.n + 2.a - n^2 - 2.n - 2)\ ]^2 \\
&- [\ z - p.m + p.l.a - p^2 l + t.(2.a.p - p^2 - 1)\ ]^2 \\
&- [\ q - x + y.(a - p - 1) + s.(2.a.p + 2.a - p^2 - 2.p - 2)\ ]^2 \\
&- [\ a^2.l^2 - l^2 + 1 - m^2\ ]^2 \\
&- [\ n + l + v - y\ ]^2 \\
&) \ .\ (k+2)
\end{aligned}
$$

Tests de primalité

- En général, on génère un nombre N aléatoire, et on teste s'il est premier
  - Comment tester si un nombre est premier ?
- Méthode naïve
  - tester si N est divisible par tous les entiers inférieurs à $\sqrt{N}$
  - pas efficace pour les grands nombres
- Autres méthodes exactes
  - Test AKS, temps polynomial

Tests de primalité

- Algos probabilistes
  - Répondent « oui » ou « non » avec une certaine probabilité p
  - Plus p est proche de 1, plus on est sûr
  - Ex : Fermat, Miller-Rabin, Solovay-Strassen

Test de Fermat

Si $1 \equiv 2^{(x-1)} \equiv 3^{(x-1)} \equiv 5^{(x-1)} \equiv 7^{(x-1)} [x]$

Alors x est « certainement » premier

De façon générale, si $a^{(x-1)} \equiv 1[x]$ avec a premier, x est certainement premier

a est appelé « témoin »

Augmenter le nombre de témoins augmente la chance de ne pas se tromper

Exponentiation modulaire

- But : calculer $a^c[b]$ de façon efficace
- Plusieurs techniques
  - Ex: méthode rapide
- Basée sur les équations suivantes :

$$a^2[b] \equiv (a[b])^2 \quad [b]$$

$$(m * n)[b] \equiv (m[b] * n[b]) \quad [b]$$

A vous de faire l'algo !

Indice : regarder la parité de l'exposant... cad pour a^n, regarder si n est pair ou impair et appliquer l'une des 2 formules en fonction.
Défense de regarder sur internet ! La solution (certes pas optimisée) tient en 4 lignes...

Factorisation

- Divisions successives, le plus simple
- GNFS, un des plus efficace (et complexe)
- Courbes elliptiques, efficace, probabiliste
- Factorisation de Fermat, simple
  - Principe : tout N impair se décompose ainsi :

$$N = a^2 - b^2 = (a+b).(a-b)$$

Factorisation de Fermat

```
FactorFermat(N): // N doit être impair
    A := partie_entiere(sqrt(N))
    Bsq := A*A - N
    tant que Bsq n'est pas un carré:
        A := A + 1
        Bsq := A*A - N
    retourner (A - sqrt(Bsq) , A + sqrt(Bsq))
```

Conclusion

- Cryptographie basée sur les maths
  - Certaines propriétés restent non prouvées
    - ex : difficulté de casser RSA
  - Importance des nombres premiers
- Sans oublier...
  - Les méthodes présentées ici sont simples, la réalité est bien plus complexe...
  - Les démos des propriétés ont été (volontairement) zappées

Références

- Wikipédia FR
- *Comprendre, implémenter et casser RSA*, dedji, Hackademy Manuel n°8