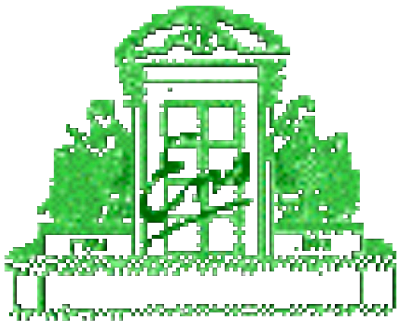
# Initiation à la cryptographie

## Factorisation, Log discret et Intégrité, Authentification

Pierre-Alain FOUQUE
Équipe de Cryptographie
École normale supérieure



# Algorithme pour la Factorisation

- Strassen – Pollard
- Méthode Rho
- Méthode p-1
  - Courbe elliptique
- Carré aléatoire
  - Crible quadratique
  - Crible de corps de nombre

# Algorithme de Strassen

- Algorithme déterministe prouvé en temps $O(N^{1/4}\log^2(N))$
- $B = N^{1/4}$ et $f(x) = x(x-1)\ldots(x-B+1)$
  - $f(jB) = (jB)!/((j-1)B)!$
- Si $\text{pgcd}(N,f(jB))>1$, alors facteur de N
  - recherche exhaustive parmi B valeurs
- Calcul de $\text{pgcd}(N,f(jB))$ pour $j=1, \ldots, B$ en temps $O(B\log^2(B))$

# Méthode Rho de Pollard

- Idée: $f:Z_N \rightarrow Z_N$ et $x_0 \in Z_N$ et $x_i = f(x_{i-1})$ $\forall i \geq 1$
- On espère que $x_0, x_1, x_2, \ldots$ sera une suite de valeurs de $Z_N$ aléatoires et indépendantes
- Si p facteur inconnu de N, on aura une collision mod p si $\exists t$ et $l>0$ t.q. $x_t = x_{t+l} \bmod p$
- Si N n'est pas premier et q un autre facteur, et $x_i$ mod N aléatoire, alors $x_i$ mod p et $x_i$ mod q sont indépendants et aléatoires (CRT)
- Improbable $x_t = x_{t+l} \bmod q$, $\text{pgcd}(x_t - x_{t+l}, N) = p$
- Méthode heuristique: $O(\sqrt{p})$ et $f(x) = x^2 + 1$
- Amélioration mémoire: Astuce de Floyd

# Méthode p-1

- N=pq tq p-1 est B-lisse
- x B-lisse si tous ces facteurs prem $\leq B$
- Soit $Q=\prod_{i|\ q\leq B} q^{\lfloor \ln N/\ln q \rfloor}$
- $(p-1)|Q$, $\forall a\in Z^*_p\ a^Q=1 \bmod p$ (Fermat)
- Alors $\text{pgcd}(a^Q-1,N)=p$
- Temps de calcul: $O(B \ln N/\ln B)$ mult. mod.
- ECF: Extension de p-1 à p-c car l'ordre de la courbe est dans l'intervalle $p+1\pm\sqrt{p}$
- Coût ECF: $L_p(1/2,1)$ où

$$L_q(\alpha,c)=O(\exp((c+o(1))(\ln q)^{\alpha}(\ln \ln q)^{1-\alpha}))$$

# Carrés aléatoires

- Si $x^2=y^2 \bmod N$ et $x \neq \pm y \bmod N$, alors pgcd(x-y,N) facteur non-trivial de N
- Choisir une base de facteur $S=\{p_1,\ldots p_t\}$
  - Trouver des paires $(a_i,b_i)$ tq
    - $a_i^2=b_i \bmod N$
    - $b_i=\prod_j p_j^{eij}$ $e_{ij} \geq 0$, ($b_i$ est $p_t$-lisse)
  - Trouver un sous-ensemble des $b_i$ dont le produit est un carré parfait (utiliser de l'algèbre creuse)
- Algorithme de Dixon:
  - $a_i$ est choisi au hasard et division par les $p_i$
- Crible quadratique:
  - $x=\lfloor \sqrt{N} \rfloor$ et $q(x)=(x+m)^2-N$. $q(x)=x^2-2mx+m^2-N \approx x^2+2mx$
  - Si $a_i=(x+m)$ avec x petit, $b_i=(x+m)^2-N$ sera petit et $p_t$-lisse avec plus forte proba. Rq: si $p_i | b_i$ alors $(N|p_i)=1$

# Algorithme pour le Log. Discret

- Baby-Step Giant-Step
- Rho de Pollard (sans mémoire)
- Pohlig-Hellman

# Baby Step Giant Step

- Compromis temps/mémoire
- Soit n ordre de $\alpha$, $m=\lceil\sqrt{n}\rceil$,
- Si $\beta=\alpha^x$, $x=i\times m+j$ où $0\leq i,j<m$ et

  $\alpha^x=\alpha^{im}\times\alpha^j$ donc, $\beta(\alpha^{-m})^i = \alpha^j$
- Faire une table avec l'ensemble des valeurs $(j,\alpha^j)$ et trier suivant $\alpha^j$
- Précalculer $\alpha^{-m}$ et matcher $\beta(\alpha^{-m})^i$
- Coût: $O(\sqrt{n})$ mult. mod. et comparaisons

# Pohlig-Hellman

- Si $n=\prod_i p_i^{ei}$, alors pour chaque pi, on peut rechercher l'ordre mod $p_i^{ei}$
- On recherche l'ordre mod $p_i$ avec la technique précédente ou directement
- $x=x_0+x_1p_i+\ldots x_ip^{ei}$ $x_i$ après l'autre $x_{i-1}$
- Finir avec CRT

# Notion de sécurité

- Pour déterminer la taille des clés à utiliser, on utilise le meilleur algorithme pour casser la primitive et à l'aide de sa complexité C, on évalue le plus petit entier n tq $C(n) \geq 2^{80}$
- Aujourd'hui, des calculs ont été faits jusqu'à $2^{64}$ opérations de chiffrement symétrique (DES par exemple) avec 40 000 machines pendant 5 ans
- $2^{80}$ est le niveau le plus bas
- $2^{128}$ pour les applications plus sensibles

# Record de Factorisation et Log Discret et Conséquences

- Factorisation:
  - RSA200: 663 bits en mai 2005 (200 chiffres décimaux) avec 80 AMD Opteron
- Log Discret:
  - Mod p: en juin 2005 (130 chiffres décimaux) avec 16 processeurs en un mois
  - Dans GF($2^n$) : n=607 et n=613 16 processeurs en 2 mois
- 1024 bits est un minimum pour RSA et DL sur $Z_p^*$
- Sur EC, on peut prendre p de 160 bits car les algorithmes pour DL sont plus lents

# Introduction: sécurité prouvée

- Historiquement, la cryptographie a été une course entre les cryptographes et les cryptanalystes
- Au début des années 1980, on a cherché à formaliser et asseoir la cryptographie sur des bases solides

# Problème des hypothèses

- On ne sait pas si RSA est équivalent à la factorisation (RSA $\leq$ Fact.)
- Schéma de Rabin:
  - $E(m)=m^2 \bmod N$ où $N=pq$
  - D(c)=CRT(rac. carrée mod p, rac. carrée mod q)
  - Par ex. si $p=q=3 \bmod 4$, $x^{1/2}=x^{(p+1)/4} \bmod p$
- Si on sait déchiffrer, alors on sait factoriser
  - Si A sait déchiffrer, construire une MT qui utilise A comme sous-routine et factorise

# Sécurité par réduction

- Prouver la sécurité d'un schéma: technique des preuves par réduction en complexité
  - Si P1 se réduit à P2, noté $P1 \leq P2$, et si P1 est difficile, alors P2 sera aussi difficile
- Réduction vers problèmes algorithmiques conjecturés comme difficile (P1)
- Preuve par l'absurde
  - On montre que s'il existe un adversaire contre le schéma cryptographique (algo A), alors il existe un attaquant (algo B) contre le problème algorithmique
  - Comme le problème algorithmique est supposé difficile, il n'existe pas d'algo B et donc pas d'algo A

# Formalisation d'une attaque

- Description d'une attaque en terme de problème algorithmique
- Adversaire: algorithme
- But de l'adversaire: ce qu'il cherche à faire
  - calculer une fonction ou son inverse sur une entrée précise
- Modèle d'attaque: les moyens qu'il a à sa disposition
  - valeurs entrées/sorties de la fonction qu'il connaît
- Dans certains cas, modéliser des fonctions difficiles à décrire en terme mathématique
  - Modèle de l'oracle aléatoire (Fonction de hachage = fonction aléatoire)

# Algorithme efficace

- Algorithme efficace: MTD probabiliste en temps et mémoire polynomial (PPT)
  - Bande supplémentaire externe qui contient des bits générés selon une distribution uniforme et indépendamment les uns des autres $r \in \{0,1\}^{t_M(x)}$
  - La sortie et le temps dépendent de la valeur r des bits aléatoires. M(x,r) et $t_M(x)$ sont des VA
  - Indépendamment de r, M s'arrête après $t_M(x)$ pas de calculs et donc r est borné

$$Pr(M(x,r)=y : r \in \{0,1\}^{tM(x)}) = |\{r \in \{0,1\}^{tM(x)} : M(x,r)=y\}| / 2^{tM(x)}$$

# Adversaire et Probabilités

- Adversaire: algo. PPT
  - Éviter qu'un adversaire déchiffre ou signe avec proba 1/2
  - On aimerait $Pr(A(y)=x : x \leftarrow Dom, y=E(x)) \leq 1/2^n$ où n est la taille de y en bits
- Probabilité :
  - Proba. *écrasante* $\geq 1-1/p(n)$ pour tout polynôme p et $n \geq N$ (par exemple $1-1/2^n$)
  - Proba. *négligeable* $\leq 1/p(n)$ pour tout polynôme p et $n \geq N$ (par exemple $= 1/2^n$)
  - En pratique, dès que proba. $\geq 1-1/2^{80}$ (écrasante) et proba. $\leq 1/2^{80}$ (négligeable)

# Sécurité pratique

- On veut que les réductions réussissent avec la meilleure proba.
  - Si on casse avec proba. $\varepsilon$ le schéma initial, alors on casse l'hypothèse algorithmique avec proba. $\varepsilon' = q\varepsilon$ (q nombre de questions par ex)
  - or $\varepsilon = 2^{61}$ pour $|N| = 768$ et donc si $q = 2^{60}$, alors la preuve nous dit que la proba de casser le schéma est 1/2 ce qui est loin d'être négligeable
  - en pratique, il faut donc choisir $\varepsilon' = 2^{80}$, et si $q = 2^{40}$, alors $\varepsilon = 2^{120}$ pour $|N| = 4096 \Rightarrow$ moins eff.
- Preuve fine: $\varepsilon' \approx \varepsilon$

# Efficacité pratique

- Pour évaluer l'efficacité d'une réduction, utiliser le rapport $T/\varepsilon$ où
  - T représente le temps de la réduction et
  - $\varepsilon$ la probabilité de réussite
- Si le temps de la réduction est trop grand, alors on augmente aussi le temps pour casser l'hypothèse
  - avec plus de temps, on peut casser des modules plus grand par exemple

# Intégrité et Authentification

- Intégrité: garantir que le contenu d'une communication ou d'un fichier n'a pas été modifié
- Authentification: garantir l'identité d'une entité (*identification*) ou l'origine d'une communication ou d'un fichier (*authentification de données*)
  - Non-répudiation (signature): le signataire ne peut pas dénier sa signature

# Identification

- Pour s'identifier une fois à Bob ou Charlie, Alice utilise une fonction à sens unique et leur envoie y=f(x)
- Pour s'identifier, Alice envoie x
- Pour se faire passer pour Alice, Charlie doit calculer x' tq f(x')=f(x)
- Si un tel adversaire existe, alors on a casser la fonction à sens unique

# Identification Multiple

- Pour s'identifier n fois auprès du même serveur, on utilise OTP (One-Time Password)
- Soit f une fonction à sens unique (fonction de hachage par exemple H),
- On choisit x aléatoire et on calcule $y_n=f^n(x)$ (n fois f), on envoie $y_n$ au serveur
- La i-ième fois, on envoie $x_{n-i}$ tq $f(x_{n-i})=y_{n-i}$ et le serveur remplace $y_{n-i}$ par $x_{n-i}$

# Signature

- Propriétés garanties:
  - Intégrité des donnés transmises
  - Authentification de l'émetteur (preuve de connaissance de la clé secrète)
  - Non-répudiation
- Similaire au MAC mais avec de la clé publique

# Schéma de signature

- $G(1^n)$: PPT algo. de génération de clé
- $S_{sk}(m)$: PPT algo. de génération de signature
- $V_{pk}(m,s)$: algo. de vérification à valeur {0,1}
  - « 1 »=signature valide, « 0 »=signature invalide
- Pour tout $m \in M_n$, $V_{pk}(m,S_{sk}(m))=1$

# Buts de l'adversaire

- Cassage total: retrouver la clé secrète
- Falsification : Un schéma de signature S est dit difficile à falsifier (contrefaire) si:
  - Falsification universelle (algo. équivalent signer)
  - Falsification sélective (falsifier certains messages déterminés à l'avance)
  - Falsification existentielle (falsifier au moins un message – exemple: RSA)

$$\Pr[A(m)=s : (sk,pk)\leftarrow G(1^n), m\leftarrow M_n, V_{pk}(m,s)=1] \leq negl(n)$$

# Modèle de sécurité

- Moyens:
  - Attaque à messages connus
  - Attaque à messages choisis générique (indépendamment de la clé)
  - Attaque à messages choisis orienté
  - Attaque à messages choisis adaptative (CMA)
- Notion de sécurité: Association entre
  - Le but de l'adversaire: ce qu'il cherche à faire pour casser le système
  - Les moyens d'y parvenir
  - Exemple: EF-CMA, …

$$\Pr[A(m)=s : (sk,pk)\leftarrow G(1^n), m\leftarrow M_n, m\neq m_i,\ s_i\leftarrow A^S(m_i), V_{pk}(m,s)=1] \leq negl(n)$$

# Signature de Lamport

- A partir de n'importe quelle OWF, schéma de signature sûr contre une seule signature
- Pour signer un message de k bits: $m=m_1 \ldots m_k$
- On génère 2k valeurs $x_i^j$ aléatoires dans le domaine d'entrée de f et les images $y_i^j=f(x_i^j)$
- Clé publique: $\{y_i^j\}_{i=1,..,k}$ pour j=0 et 1
- Clé secrète: $\{x_i^j\}_{i=1,..,k}$ pour j=0 et 1
- Signature: $x_1^{m_1} x_2^{m_2} \ldots x_k^{m_k}$

Th: Si on sait casser signature avec proba $\varepsilon$, on sait inverser f avec proba $\varepsilon/2$

# Signature padding fixe

- RSA PKCS#1 v1.5 avec padding fixe
- x=00||FFFFFFFFFFFFFFFFFFFF||H(M) où H est une fonction sur 160 bits et $s=x^d \bmod N$
- Si on casse la fonction de hachage, alors on peut casser EF-CMA

# Signature RSA

- $f(x)=x^e \bmod N$ et son inverse $g(x)=x^d \bmod N$
- Clé publique: f ou de manière équivalente (e,N)
- Clé secrète: g ou (d,N)
- H une fonction de hachage de $\{0,1\}^*$ vers $Z_N$
- $S(m)=g(H(m))=s$
- $V(m,s)=(f(s)==H(m))$
- Si $H:\{0,1\}^* \rightarrow Z_N$, le schéma est appelé FDH (Full Domain Hash)

Rem: Si on ne hache pas m, falsification existentielle ($m=s^e \bmod N$, s) est un couple message-signature valide !!!

# Sécurité

Th: Dans le modèle de l'oracle aléatoire, RSA-FDH est EF-CMA

- Preuve: Supposons qu'il existe un adversaire F contre le schéma RSA-FDH avec proba. non négligeable $\lambda(n)$, on peut construire un adversaire I qui inverse RSA avec proba.

$$\varepsilon(n) > \lambda(n)/(q_H + q_S)$$

où $q_H$ est le nombre de questions à l'oracle H et $q_S$ le nombre à l'oracle de signature

# Sécurité

Th: Dans le modèle de l'oracle aléatoire, le schéma RSA-FDH est EF-CMA

- Preuve: Supposons qu'il existe un adversaire F contre le schéma RSA-FDH proba. non négligeable $\lambda(n)$, on peut alors construire un adversaire I qui inverse RSA avec proba.

  $\varepsilon(n) > \lambda(n)/q_S(n)$ où $q_S(n)$ est le nombre de questions à l'oracle S

# Schémas de signature basés sur des preuves ZK

- Les preuves ZK permettent de construire des schémas d'authentification
- Prouver la connaissance d'une clé secrète
- Les signatures garantissent en plus l'intégrité d'un document et la preuve d'authentification est attachée au msg

# Heuristique Fiat-Shamir
# Ex. Signature Schnorr

- Soit p un nombre premier, q un grand diviseur premier de p-1 et g un générateur du sous-groupe d'ordre q
- P a une clé secrète x et une clé publique $y=g^x \bmod p$
- Signature: choisir $k \leftarrow Z_q$, et $r=g^k \bmod p$
- $c=H(p,q,g,r,m)$
- et $s=k+xc \bmod q$
- Signature: (m,(c,s))
- Vérification: $H(p,q,g, g^s \times y^{-c},m)=c$

# Schéma de signature DSA

- Soit p un nombre premier, q un grand diviseur premier de p-1 et g un générateur du sous-groupe d'ordre q
- P a une clé secrète x et une clé publique $y=g^x \bmod p$
- Signataire: $k \leftarrow Z_q$, et $r=(g^k \bmod p) \bmod q$
- $s=(xr+H(m))/k \bmod q$
- Signature: (m,(r,s))
- Vérification:$u=H(m)/s$ et $v=r/s$ et vérifie si $r==(g^u y^v \bmod p) \bmod q$

# Signature basée sur DDH

- G un groupe d'ordre q premier
- PK=(g,h,$y_1$,$y_2$) tq $y_1=g^x \in G$ et $y_2=h^x \in G$
- SK=x
- Sign(m)=(c,s) tq $c=H(PK,A,B,m) \in Z_q$, $s=cx+r \bmod q$
- Ver(m,(c,s))=1 si c==H(PK,A',B',m) où $A'=g^s y_1^{-c}$ et $B'=h^s y_2^{-c}$

# Conclusion

- Les preuves par réduction ne sont pas des preuves absolues (modulo l'hypothèse algorithmique)
- Elles donnent cependant confiance dans le schéma s'il n'y a pas d'attaque dans ce modèle
- Est-ce que le modèle couvre toutes les attaques ?