Chapitre 4 - Algorithmes à clés privée/publique

Alexandra Bruasse-Bac

Plan

- Introduction
- Le système RSA
- Le système de Rabin
- Le système ElGamal
- Les système McEliece, Knapsack, les systèmes probabilistes
- Et après ...

Introduction

Problème de l’infratstructure de gestion des clés pour des algorithmes de chiffrement symétrique.

- une clé par paire d’utilisateurs

Problème de l'infratstructure de gestion des clés pour des algorithmes de chiffrement symétrique.

- une clé par paire d'utilisateurs
- n utilisateurs : $n(n-1)/2$ clés

Problème de l'infratstructure de gestion des clés pour des algorithmes de chiffrement symétrique.

- une clé par paire d'utilisateurs
- n utilisateurs : $n(n-1)/2$ clés

- Echange de ces clés
- Autorité centrale de gestion des clés

Problème de l'infratstructure de gestion des clés pour des algorithmes de chiffrement symétrique.

- une clé par paire d'utilisateurs
- n utilisateurs : $n(n-1)/2$ clés

- Echange de ces clés
- Autorité centrale de gestion des clés
  $\rightarrow$ accède à tous les messages

Algorithme à clé publique/privée :

- Paire de clé de codage/décodage (d, e) distinctes
- d incalculable à partir de e

Algorithme à clé publique/privée :

- Paire de clé de codage/décodage (d, e) distinctes
- d incalculable à partir de e
- Clé publique : e
- Clé privée : d

Algorithme à clé publique/privée :

- Paire de clé de codage/décodage (d, e) distinctes
- d incalculable à partir de e
- Clé publique : e
- Clé privée : d

Besoins connexes :

- authentification de la clé publique
- authentification et intégrité des données

Le système RSA

RSA (R. Rivest, A. Shamir, L. Adleman)
Sécurité basée sur la *factorisation des entiers*

Génération des clés

(i) Générer deux grands nombres *premiers* p et q

(ii) Soit $n = pq$ et $\varphi(n) = (p-1)(q-1)$

(iii) Soit e un entier tel que

$$e \wedge \varphi(n) = 1$$

(iv) Calculer (Euclide étendu) l'inverse d de e :

$$ed \equiv 1 \quad [\varphi(n)]$$

RSA (R. Rivest, A. Shamir, L. Adleman)
Sécurité basée sur la *factorisation des entiers*

Cryptage

(i) Soit (n, e) une clé publique

(ii) Représenter le message par $m \in [0 \dots n-1]$

(iii) Soit

$$c = m^e \quad [n]$$

c est le texte crypté

RSA (R. Rivest, A. Shamir, L. Adleman)
Sécurité basée sur la *factorisation des entiers*

Décryptage

(i) Soit d la clé privée associée à (n, e)

(ii) Décryptage :

$$m = c^d \quad [n]$$

Sécurité de RSA

Proposition

Calculer d à partir de (n, e) est équivalent à factoriser n

Proposition

Calculer d à partir de (n, e) est équivalent à factoriser n

$\Rightarrow$ Si on sait factoriser n : comme au cryptage

Proposition

Calculer d à partir de (n, e) est équivalent à factoriser n

$\Rightarrow$ Si on sait factoriser n : comme au cryptage

$\Rightarrow$ Si on sait calculer $d = e^{-1}\ [\varphi(n)]$ à partir de n et e

factorisation de n ...

Proposition

Calculer d à partir de (n, e) est équivalent à factoriser n

$\Rightarrow$ Si on sait factoriser n : comme au cryptage

$\Rightarrow$ Si on sait calculer $d = e^{-1}\ [\varphi(n)]$ à partir de n et e

factorisation de n ...

$\rightarrow$ Pas de preuve que casser RSA est équivalent à factoriser n.

Attaque des petites exposants

- Pour accélérer le cryptage : e petit (supposons $e = 3$)
- Si A envoie m à plusieurs entités avec des clés

$$(e, n_1), \ldots, (e, n_i)$$

Attaque des petites exposants

- Pour accélérer le cryptage : e petit (supposons $e = 3$)
- Si A envoie m à plusieurs entités avec des clés

$$(e, n_1), \ldots, (e, n_i)$$

$$\begin{cases} x & \equiv & c_1 & [n_1] \\ x & \equiv & c_2 & [n_2] \\ x & \equiv & c_3 & [n_3] \end{cases}$$

Attaque des petites exposants

- Pour accélérer le cryptage : e petit (supposons $e = 3$)
- Si A envoie m à plusieurs entités avec des clés

$$(e, n_1), \ldots, (e, n_i)$$

$$\left\{ \begin{array}{lll} x & \equiv & c_1 \quad [n_1] \\ x & \equiv & c_2 \quad [n_2] \\ x & \equiv & c_3 \quad [n_3] \end{array} \right.$$

Comme $m^3 < n_1 n_2 n_3$ on a $x = m^3$ et m est la racine cubique de x.

Attaque des petites exposants

- Pour accélérer le cryptage : e petit (supposons $e = 3$)
- Si A envoie m à plusieurs entités avec des clés

$$(e, n_1), \ldots, (e, n_i)$$

$$\begin{cases} x & \equiv & c_1 & [n_1] \\ x & \equiv & c_2 & [n_2] \\ x & \equiv & c_3 & [n_3] \end{cases}$$

$\rightarrow$ éviter e trop petit

Multiplicativité

Si m_1 et m_2 sont deux messages cryptés en c_1 et c_2 :

$$(m_1 m_2)^e = m_1^e m_2^e \equiv c_1 c_2 \quad [n]$$

Multiplicativité

Si m_1 et m_2 sont deux messages cryptés en c_1 et c_2 :

$$(m_1 m_2)^e = m_1^e m_2^e \equiv c_1 c_2 \quad [n]$$

Adversaire veut décrypter $c = m^e \quad [n]$ destiné à A

$\rightarrow$ Si A décrypte les messages différents de c

Multiplicativité

Si m_1 et m_2 sont deux messages cryptés en c_1 et c_2 :

$$(m_1 m_2)^e = m_1^e m_2^e \equiv c_1 c_2 \quad [n]$$

Adversaire veut décrypter $c = m^e \quad [n]$ destiné à A

$\rightarrow$ Si A décrypte les messages différents de c

$\rightarrow$ Présenter au décryptage $\bar{c} = x^e c \quad [n]$ avec x aléatoire

Multiplicativité

Si m_1 et m_2 sont deux messages cryptés en c_1 et c_2 :

$$(m_1 m_2)^e = m_1^e m_2^e \equiv c_1 c_2 \quad [n]$$

Adversaire veut décrypter $c = m^e \quad [n]$ destiné à A

$\rightarrow$ Si A décrypte les messages différents de c

$\rightarrow$ Présenter au décryptage $\bar{c} = x^e c \quad [n]$ avec x aléatoire

$\rightarrow$ Ajouter des bits fixes aux messages

Accélération de RSA

En utilisant le théorème des restes chinois.

En utilisant le théorème des restes chinois.

Décryptage de c :

- Calculer :

$$m_p = c^{d \mod (p-1)} \mod p \qquad m_q = c^{d \mod (q-1)} \mod q$$

En utilisant le théorème des restes chinois.

Décryptage de c :

- Calculer :

$$m_p = c^{d \mod (p-1)} \mod p \qquad m_q = c^{d \mod (q-1)} \mod q$$

Théorème des restes chinois

$$\begin{cases} m \equiv m_p & \mod p \\ m \equiv m_q & \mod q \end{cases}$$

m unique modulo $n = pq$

Choix de p et q

- factorisation par les courbes elliptiques : p et q de même longueur, assez grands
- factorisation par la méthode $p-1$ de Pollard ou le crible quadratique : p et q entiers premiers forts

p est un entier premier fort si :

(i) $p-1$ a un grand facteur premier r

(ii) $p+1$ a un grand facteur premier

(iii) $r-1$ a un grand facteur premier

Recommendations

- Module n de 512 bits minimum (1024 pour une sécurité optimale)
- Pour des questions d'efficacité :

$\rightarrow$ e petit (souvent 3)

$\rightarrow$ ou e a peu de 1 dans son développement binaire

Recommendations

- Module n de 512 bits minimum (1024 pour une sécurité optimale)
- Pour des questions d'efficacité :

$\longrightarrow$ e petit (souvent 3)

cryptage rapide (une multiplication, un carré modulaires)

$p-1$, $q-1$ ne doivent pas être divisibles par 3

$\longrightarrow$ ou e a peu de 1 dans son développement binaire

Recommendations

- Module n de 512 bits minimum (1024 pour une sécurité optimale)
- Pour des questions d'efficacité :

$\longrightarrow$ e petit (souvent 3)

$\longrightarrow$ ou e a peu de 1 dans son développement binaire

$e = 2^{16} + 1$

cryptage : une multiplication, 16 carrés modulaires

resiste à l'attaque des petits exposants

Le système de Rabin

Cryptage / Décryptage

Cryptosystème de Rabin

Cryptage basé sur la factorisation des entiers.

Le décryptage est prouvé être équivalent au problème de la factorisation des entiers.

Cryptosystème de Rabin

Génération de clés

(i) Générer deux grands premiers p et q distincts.

(ii) Soit $n = pq$.

Clé publique n / Clé privée (p, q)

Cryptosystème de Rabin

Cryptage

(i) Soit n une clé publique.

(ii) Représenter le message par $m \in [0 \ldots n-1]$.

(iii) Soit

$$c = m^2 \quad [n]$$

c est le texte crypté

Cryptosystème de Rabin

Décryptage

(i) Calculer les 4 racines m_1, m_2, m_3, m_4 de c modulo n (grâce à p et q)

(ii) Procédure de décision

Racines carrées dans $\mathbb{Z}/n\mathbb{Z}$

Racines carrées modulo $n = pq$ (cas général)

Soit a un entier et $n = pq$ avec p, q premiers.

(i) Calculer les racines r et $-r$ de c modulo p

(ii) Calculer les racines s et $-s$ de c modulo q

(iii) Algorithme d'Euclide étendu :

$$ap + bq = 1$$

(iv) $x = rbq + sap$ et $y = rbq - sap$

$$\{\pm x, \pm y\}$$

Racines carrées dans $\mathbb{Z}/n\mathbb{Z}$

Cas où $p \equiv q \equiv 3 \ [4]$

(i) Algorithme d'Euclide étendu : $ap + bq = 1$

(ii) Calculer $r = c^{\frac{(p+1)}{4}} \ \ [p]$

(iii) Calculer $s = c^{\frac{(q+1)}{4}} \ \ [q]$

(iv) Calculer $x = aps + bqr \ \ [n]$

(v) Calculer $y = aps - bqr \ \ [n]$

Les racines sont :

$$x, -x, y, -y \ \ [n]$$

Sensibilité à une attaque par texte crypté choisi

Sensibilité à une attaque par texte crypté choisi

Comment éviter cela / choisir parmi les 4 racines ?

- Ajouter des redondances prédéterminées

Sensibilité à une attaque par texte crypté choisi

Comment éviter cela / choisir parmi les 4 racines ?

- Ajouter des redondances prédéterminées
- Probabilité élevée qu'une seule des quatre racines ait cette redondance

Sensibilité à une attaque par texte crypté choisi

Comment éviter cela / choisir parmi les 4 racines ?

- Ajouter des redondances prédéterminées
- Probabilité élevée qu'une seule des quatre racines ait cette redondance
- Protection contre l'attaque par texte crypté choisi

Sensibilité à une attaque par texte crypté choisi

Comment éviter cela / choisir parmi les 4 racines ?

- Ajouter des redondances prédéterminées
- Probabilité élevée qu'une seule des quatre racines ait cette redondance
- Protection contre l'attaque par texte crypté choisi
- Preuve d'équivalence avec le factorisation plus valable

Efficacité

- Cryptage très efficace (un seul carré modulaire)
- Décryptage plus lent mais comparable à celui de RSA

Le système ElGamal

Cryptosystème El Gamal

Securité basée sur le problème du logarithme discret et sur le schéma de Diffie-Hellman.

Cryptosystème El Gamal

Génération des clés

(i) Générer un grand premier p et un générateur α de $\mathbb{Z}/p\mathbb{Z}^*$

(ii) Choisir $a \in \{1, \ldots, p-2\}$ et calculer $\alpha^a \quad [p]$

Clé publique (p, α, α^a) / Clé privée a

Cryptosystème El Gamal

Cryptage

(i) Soit (p, α, α^a) une clé publique

(ii) Représenter le message par $m \in \{1 \ldots p-1\}$

(iii) Soit $k \in \{1, \ldots, p-2\}$ aléatoire

(iv) Calculer $\gamma = \alpha^k \ \ [p]$ et $\delta = m \cdot (\alpha^a)^k \ \ [p]$

$c = (\gamma, \delta)$ est le texte crypté

Cryptosystème El Gamal

Décryptage

(i) Utiliser a pour calculer

$$\gamma^{p-1-a} = \gamma^{-a} \quad [p]$$

(ii) Calculer

$$m = (\gamma^{-a}) \cdot \delta \quad [p]$$

- Possible que tous les systèmes conviennent d'un premier p et d'un générateur α

  $\rightarrow$ clé publique $\alpha^a \ [p]$

Sécurité, efficacité

- Possible que tous les systèmes conviennent d'un premier p et d'un générateur α

  $\longrightarrow$ clé publique $\alpha^a \ \ [p]$
- Depuis 1996, module p de 768 bits recommandé - pour la sécurité à long terme : 1024 et plus

Sécurité, efficacité

- Possible que tous les systèmes conviennent d'un premier p et d'un générateur α

  $\rightarrow$ clé publique $\alpha^a \quad [p]$

- Depuis 1996, module p de 768 bits recommandé - pour la sécurité à long terme : 1024 et plus

- Très important que k soit *aléatoire*

  k utilisé pour crypter m_1 et m_2 (en (γ_1, δ_1) et (γ_2, δ_2))

  alors $\delta_1/\delta_2 = m_1/m_2$ et on retrouve m_2 a partir de m_1

- Possible que tous les systèmes conviennent d'un premier p et d'un générateur α

  $\rightarrow$ clé publique $\alpha^a \quad [p]$
- Depuis 1996, module p de 768 bits recommandé - pour la sécurité à long terme : 1024 et plus
- Très important que k soit *aléatoire*
- Inconvénient : augmentation de la taille du message d'un facteur 2

ElGamal généralisé

On a travaillé dans $\mathbb{Z}/p\mathbb{Z}^*$, mais en fait :

complexité du logarithme discret dans un groupe G

ElGamal généralisé

On a travaillé dans $\mathbb{Z}/p\mathbb{Z}^*$, mais en fait :

complexité du logarithme discret dans un groupe G

Deux critères :

- efficacité
- sécurité

On a travaillé dans $\mathbb{Z}/p\mathbb{Z}^*$, mais en fait :

complexité du logarithme discret dans un groupe G

Les groupes suivants satisfont ces conditions :

- $\mathbb{F}_p^*$
- $\mathbb{F}_{2^m}^*$
- Groupe des points d'une courbe elliptique sur un corps fini
- $\mathbb{F}_{p^m}^*$
- ...

Le système McEliece

Le cryptosystème McEliece

Basé sur les codes correcteurs d'erreurs.

$\longrightarrow$ utiliser un code particulier (décodage efficace) pour crypter

$\longrightarrow$ le déguiser en un code linéaire quelconque

$\longrightarrow$ décodage d'un code linéaire quelconque NP-complet

Clé publique : description du code linéaire

Clé privée : description du code original

Le cryptosystème McEliece

- A résisté à la cryptanalyse jusqu'à aujourd'hui
- Très efficaces
- Peu d'attention à cause de la taille des clés publiques. Pour les paramètres recommandés :
  - 2^{19} bits
  - facteur d'expansion du message : 1,6

Le système Knapsack

Le cryptosystème Knapsack

Basé sur le problème du subset sum qui est NP-complet.

Idée similaire à celle de McEliece :

- $\longrightarrow$ utiliser un problème particulier de subset sum pour crypter
- $\longrightarrow$ le déguiser en un problème quelconque
- $\longrightarrow$ résolution d'un subset sum quelconque NP-complet

Clé publique : description du problème de subset sum
Clé privée : description du problème original (cas particulier)

Le cryptosystème Knapsack

- De nombreux schémas (cas particuliers de subset sum)
- Premier système de cryptage à clé publique
- S'avèrent tous non sûrs (sauf Chor-Rivest knapsack)
- Efficacité du schéma de Chor-Rivest :
  - Cryptage très rapide
  - Décryptage beaucoup plus lent
  - Clé publique très lourde (pour les paramètres conseillés : 36000 bits ...)

Conclusion

Conclusion

Comparaisons, produits commerciaux ....