CRYPTOGRAPHIE

Chapitre IV : Cryptographie à clés publiques
Licence Réseaux Système et Sécurité

Mr. Ahmed KHALIFA
khalifaahmedou@yahoo.fr

Cryptographie à clés publiques

Sommaire

6.1 : Préliminaires

Préliminaires

Types de cryptosystèmes à clé publique

Les cryptosystèmes clé publique se subdivisent en deux catégories : les **cryptosystèmes déterministes** et les **cryptosystèmes probabilistes**.

- Un chiffrement est déterministe (= application) si un message donné est toujours chiffré de la même manière.

Donc, on a un chiffrement déterministe si on parvient construire une **application** inversible droite (donc injective) et dont l'inverse est difficile calculer pour ceux qui ne connaissent pas la trappe c'est à dire qu'il soit une fonction à sens unique avec trappe.

6.1 : Préliminaires

Types de cryptosystèmes à clés publiques

- Un chiffrement est probabliste s'il n'est pas déterministe, plus précisément si la probabilité qu'un message soit chiffré de la même manière est très faible.

Par exemple, si on a un famille finie ou dénombrable de fontions $F = \{f_i, i \in \Lambda, \varnothing \neq \Lambda \subseteq \mathbb{N}\ f_i : \mathcal{P} \rightarrow \mathcal{D} \supseteq \mathcal{C}\}$ ayant le même inverse gauche g, on peut construire un cryptosystème probabiliste de la façon suivante :

Préliminaires

Types de cryptosystèmes à clés publiques

-1) on définie un générateur aléatoire d'entiers sur Λ : $R(\Lambda)$
-2) pour chiffrer :

- on choisie aléatoirement une fontion de F via $R(\Lambda)$ qu'on note $f_{R(\Lambda)}$
- on chiffre un message $m \in \mathcal{P}$, en calculant $f_{R(\Lambda)}(m)$
- quelque soit la valeur de $R(\Lambda)$, on déchiffre en calculant $gf_{R(\Lambda)}(m)$

Propriétés

Propriétés recherchées pour un cryptosystème clé publique :

- être probabiliste si possible ;
- être simple à comprendre et à mettre en oeuvre ;
- être éfficient (= rapide, facile à implmenter sur différents supports, consomme peu de ressources mémoires ...) ;
- être résistant contre toutes les attaques connues ;
- avoir une sécurité sémantique (= montrer que l'algorithme est cassé si et seulement si le problème difficile auquel il est relié, est résolu) (**NB** Si c'est le cas on a plus besoins de vérifier le point précédent.)
- être adapté de multiples applications (posséder la propriété d'homomorphie, être probabiliste, ... ;)

Préliminaires

Propriétés)

Propriété d'homomorphie

On note $E : \mathcal{P} \to \mathcal{C}$ un opérateur de chiffrement et $D : \mathcal{C} \to \mathcal{C}$ l'opérateur de déchiffrement associé : $D \circ E = id_{\mathcal{P}}$. On suppose que $(\mathcal{P}, \star)$ et $(\mathcal{C}, \times)$ sont deux semi-groupes commutatifs non nécessairement unitaires.

1er cas : : On dit que E possède la propriété d'homomorphie si E est un homomorphisme de sémi-groupes i.e

- (α) : $E(m \star m') = E(m) \times E(m')$ pour tout $m, m' \in \mathcal{P}$;
- on voit que $(\alpha) \Rightarrow (\beta)$:
- (β) : $D(E(m) \times E(m')) = m \star m'$ pour tout $m, m' \in \mathcal{P}$;
- mais la réciproque n'est pas forcément vraie.

Préliminaires

Propriétés (suite)

Propriété d'homomorphie (suite)

- **2me cas : Généralisation niveau 1**
  On dit que E possède la propriété d'homomorphie si
  - (β) : $D(E(m) \times E(m')) = m \star m'$ pour tout $m, m' \in \mathcal{P}$ est vérifié.
- **3me cas : Généralisation niveau 2**
  On dit que E possède la propiriété d'homomorphie si la probabilité pour que
  - $D(E(m) \times E(m')) \neq (m \star m')$ pour tout $m, m' \in \mathcal{P}$ est très faible.

Propriétés (suite)

Propriété d'homomorphie (suite)

La propriété d'homomorphie, de par ses multiples applications est une proriété très recherchée en cryptographie clés publique car elle permet essentiellement de faire des calculs sur les chiffrés : Calcul sur les chiffrés, Partage de secret, Mix- Net, Watermarking, Vote électronique, Lotterie électronique.... ,

RSA

Sommaire

Présentation

- Du nom de ses inventeurs (Rivest, Shamir et Adleman), RSA est le premier algorithme à clé publique inventé en 1978.
- Le système RSA repose sur la difficulté de factoriser de grands nombres premiers.
- Pour assurer la sécurité les nombres premiers choisis doivent être suffisamment grands.

RSA

Génération des clés

- On choisit un entier n(modulo) tel que $n = pq$ avec p et q deux nombres premiers assez grands ;
- On calcule $\varphi(n) = (p-1)(q-1)$. φ est l'indicateur d'Euler et $\varphi(n)$ le nombre d'éléments inversibles du groupe $(\frac{\mathbb{Z}}{n\mathbb{Z}})^*$;
- On choisit e tel qu'il soit plus petit que $\varphi(n)$ et $\text{pgcd}(e, \varphi(n)) = 1$;
- On calcule d tel que $ed = 1 \text{mod } \varphi(n)$ (avec l'algorithme étendu d'Euclide).
- Clé publique (e, n) : (généralement $e = 2^{16} + 1$ est fixe) ;
- Clé privée (d, n). p, q et $\varphi(n)$ doivent rester secret.

RSA

Chiffrement

Si $M \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ représentant un message, alors pour le chiffrer, il suffit de l'élever à la puissance e, le reste modulo n représente le message une fois chiffré. Nous aurons donc la représentation suivante si C est le chiffré :

$$C = M^e \pmod{n}$$

RSA

Déchiffrement

- On déchiffre un message chiffré $C \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ par :
$$C^d \mod n$$
- Puisse que d est de très grande taille, on utilise aussi l'algorithme de calcul de puissance rapide.

RSA

Signature

Le cryptosystème RSA s'utilise aussi pour générer des signatures électroniques. Une signature électronique garantit l'authenticité, l'intégrité et la non-répudiation d'un document électronique. Pour signer un message $M \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ avec RSA, le signataire utilise sa clé privée d pour obtenir la signature :

$$S = M^d \mod n$$

Vérification

Seul le signataire possédant la clé privée d est donc capable de signer le message M. On vérifie la validité de la signature S en utilisant la clé publique (e, n), en s'assurant que :

$$M = S^e \mod n$$

RSA

Forces

On peut citer entre autres :

- Chiffrement et signature en même temps ;
- Simplicité et efficience ;
- Capacité d'adaptation : toutes les attaques connues jusqu'ici ont pu être contournées :
  - soit en utilisant OAEP pour le chiffrement et PSS pour la signature pour les rendre probabilistes ;
  - soit en modifiant la taille des paramètres : n, p, q, e et d ;

RSA

Faiblesses

On peut citer entre autres :

- Son déterminisme (cause de beaucoup de vulnérabilités) ;
- Son algorithme de génération de clés :$ed = 1 \bmod \varphi(n)$;
- Ressemblance des fonctions de déchiffrement et signature ;
- La lenteur des opérations de déchiffrement et de signature ;
- La difficulté qu'on a à traiter RSA sur différents groupes finis ;
- Sa propriété d'homomorphie ;
- etc.

ELGAMAL

Sommaire

ELGAMAL

Présentation

Le chiffrement d'El Gamal est un système de cryptage à clé publique inventé en 1984 et utilisé pour chiffrer mais aussi pour signer des messages. Ce système est assez peu utilisé en tant qu'algorithme de chiffrement pur, mais plutôt dans les systèmes de signatures ; il est d'ailleurs à l'origine du DSS (Digital Signature Standard), devenu une norme fédérale en 1994. Ce chiffrement tire son nom, comme RSA, du nom de son créateur.

ELGAMAL

Génération des clés

- Choisir un nombre premier très grand p ;
- Choisir g un générateur de $(\frac{\mathbb{Z}}{p\mathbb{Z}})^*$ d'ordre $n = p - 1$;
- Alice choisit a aléatoire dans $[2, n]$, et calcule $h = g^a (\mod p)$;
- La clé publique est (g, h, p) ;
- La clé secrète est a.

ELGAMAL

Chiffrement

- Clé publique est (g, h, p) ;
- Choisir un nombre aléatoire k assez grand ;
- Pour chiffrer un message clair m, Bob calcule :
  - $m_1 = g^k \mod p$;
  - $m_2 = m.h^k \mod p$.
- Chiffré $c = (m_1, m_2)$.

ELGAMAL

Déchiffrement

- Clé privée a ;
- Alice reçoit le chiffré $c = (m_1, m_2)$;
- Elle calcule :

$$m = m_1^{p-1-a}.m_2 = g^{-ka}.g^{ka}.m = m.$$

ELGAMAL

Signature

- Alice souhaite signer un document M ;
- Elle choisit au hasard un entier $k \in [1, p-2]$; tel que $\text{pgcd}(k, p-1) = 1 \iff k^{-1} \in \mathbb{Z}_{P-1}$ existe ;
- Signature de M : $s(M) = (r, s)$ avec
  - $r = g^k \mod p$;
  - $s = k^{-1}(M - a.r) \mod (p-1)$;
- Le document signé est alors $[M, s(M)]$.

Vérification

La signature est dite valide si :

- $0 < r < p$;
- $h^r r^s \equiv g^M (\mod p)$

Forces

La force du chiffrement d'El Gamal est son usage d'un problème mathématique connu comme difficile à résoudre, celui du logarithme discret. En effet, le calcul d'un logarithme modulo n est possible mais plus n est grand et plus le calcul sera long, pouvant atteindre plusieurs milliers d'années. On définit le chiffrement dans le corps fini $\mathbb{Z}_p$, avec p un nombre premier. Le groupe $\mathbb{Z}_p^*$ est cyclique et on appelle ses générateurs racines primitives modulo p.

ELGAMAL

Faiblesses

Du nom de son créateur, c'est un autre algorithme basé sur le protocole Diffie Helmann qu'on retrouve notamment dans les dernières versions de PGP ; algorithme qui n'est pas sous brevet. A noter que le message chiffré est deux fois plus long que le message d'origine : ce qui peut engendrer quelques complications en terme de stockage ou de communication. Certaines implémentations d'ElGamal génèrent parfois des risques quant à l'exposition possible de la clé privée.

RABIN

Sommaire

Présentation

Le cryptosystème de Rabin est un cryptosystème asymétrique basé sur la difficulté du problème de la factorisation (comme RSA). Il a été inventé en 1979 par Michael Rabin : c'est le premier cryptosystème asymétrique dont la sécurité se réduit à l'intractabilité de la factorisation d'un nombre entier.

RABIN

Génération des clés

Comme pour tous les algorithmes de cryptographie asymétrique, le cryptosystème de Rabin fait usage d'une clé publique et d'une clé privée. La clé publique est utilisée pour chiffrer et n'est pas secrète, tandis que la clé privée est secrète et ne doit être connue que de son propriétaire : le destinataire du message (afin qu'il soit le seul à pouvoir décrypter). Explicitement, la génération de clés est comme suit.

- Choisisr deux grands nombres premiers, p et q, au hasard ;
- Posons $n = pq$, ce qui fait de n la clé publique. Les nombres premiers p et q constituent la clé privée ;

Pour chiffer, on n'a besoins que de la clé publique, n. Pour déchiffrer, les facteurs de n, p et q, sont nécessaires.

Chiffrement

Pour le chiffrement, seulement la clé publique, n, est utilisée.
On produit le texte chiffré à partir du texte en clair m comme suit.
Le texte chiffré c se détermine comme suit.

$$c = m^2 \mod n.$$

Autrement dit, c est le résidu quadratique du carré du texte en clair, pris modulo n. En pratique du chiffrement par bloc est généralement utilisé.

RABIN

Déchiffrement

Pour déchiffrer, la clé privée est nécessaire. Le processus est comme suit.
Les racines carrées

$$m_p = \sqrt{c} \mod p$$

et

$$m_q = \sqrt{c} \mod q$$

sont calculées.
L'algorithme d'Euclide étendu permet de calculer y_p et y_q, tels que

$$y_p \cdot p + y_q \cdot q = 1.$$

RABIN

Déchiffrement

On invoque alors le théorème des restes chinois pour calculer les quatre racines carrées $+r, -r, +s$ et $-s$ de $c + n\mathbb{Z} \in \mathbb{Z}/n\mathbb{Z}$.
($\mathbb{Z}/n\mathbb{Z}$ est l'ensemble de la classe des restes modulo n ; les quatre racines carrées sont dans l'ensemble $\{0, ..., n-1\}$) :

$$\begin{aligned} r &= (y_p \cdot p \cdot m_q + y_q \cdot q \cdot m_p) \mod n \\ -r &= n - r \\ s &= (y_p \cdot p \cdot m_q - y_q \cdot q \cdot m_p) \mod n \\ -s &= n - s \end{aligned}$$

RABIN

Génération de clés pour la signature

- Le signataire choisit deux nombres premiers $p,\ q$, chacun de taille d'environ $k/2$ bits, et calcule le produit $n = pq$;
- Il choisit au hasard b dans $\{1, ..., n\}$;
- La clé publique est (n, b) ;
- La clé privée est (p, q).

RABIN

Signature

- Pour signer un message m, le signataire choisit U un padding aléatoire et calcule $H(mU)$;
- Il résout alors $x(x+b) = H(mU) \mod n$;
- S'il n'y a pas de solution il reprend un nouveau padding U et essaie de nouveau. Si H est vraiment aléatoire le nombre de tests est 4 ;
- La signature de m est la paire (U, x).

Vérification

Compte tenu d'un message m et une signature (U, x), le vérificateur calcule $x(x+b)$ et $H(mU)$ et vérifie qu'elles sont égales

RABIN

Forces

Le cryptosystème de Rabin a l'avantage de disposer d'une preuve de difficulté aussi grande que la factorisation d'entiers, preuve qui n'existe pas encore pour RSA.

Faiblesses

Il a par contre un incovenient dû à un non-déterminisme : une sortie produite par la fonction présente dans le cryptosystème peut être le résultat de quatre entrées distinctes. Il faut donc déterminer quelle entrée est la bonne par un mécanisme annexe.

Goldwasser-Micali

Goldwasser-Micali

Sommaire

Goldwasser-Micali

Présentation

- Du nom de ses inventeurs (Shafi-Goldwasser, Micali), Goldwasser-Micali est un algorithme probabiliste à publique inventé en 1986.
- Le système Goldwasser-Micali repose sur la difficulté de la residuosité quadratique.
- Pour assurer la sécurité les nombres premiers choisis doivent être suffisamment grands et aléatoires.

Détermination d'un pseudo-carré y

$\Rightarrow$ On rappelle

$$\left(\frac{a}{p}\right)_{jac} = \begin{cases} 1, & \text{Si } a \text{ est un carré } \textit{modulo } p\,; \\ -1, & \text{Si } a \text{ n'est pas un carré } \textit{modulo } p\,; \\ 0, & \text{Si } a \text{ est divisible par } p. \end{cases},$$

$\Rightarrow$: $\left(\frac{a}{p}\right)_{jac} = a^{\frac{p-1}{2}} \, mod \ p$; $\left(\frac{a}{pq}\right)_{jac} = \left(\frac{a}{p}\right)_{jac}\left(\frac{a}{q}\right)_{jac}$.

$\Rightarrow$ **Problème de residuosite quadratique : Soit n un entier produit de deux entiers aléatoires assez grands et a un entier de symbole de Jacobi 1 c'est à dire $\left(\frac{a}{n}\right)_{jac} = 1$; decider si a est residu quadratique ou non.**

Goldwasser-Micali

Détermination d'un pseudo-carré y

$\Rightarrow$ Choisir $a,\ b$ aléatoires tels que $\left(\frac{a}{p}\right)_{jac} = -1$ et $\left(\frac{b}{q}\right)_{jac} = -1$;
$\Rightarrow$ Comme la probabilité de tomber sur un residu non quadratique est de $\frac{1}{2}$ alors la recherche aboutit vite ou bien on le construit en prend un générateur α de $(\frac{\mathbb{Z}}{p\mathbb{Z}})^*$ et un exposant pair aléatoire $2k$ puis poser $a = \alpha^{2k} mod\ p$.
$\Rightarrow$ On utilise ensuite le Theoreme des Restes Chinoix pour calculer y tel que $y = a\ mod\ p$ et $y = b\ mod\ q$. Alors
$\left(\frac{y}{n}\right)_{jac} = \left(\frac{y}{p}\right)_{jac}\left(\frac{y}{q}\right)_{jac} = (-1)(-1) = 1$

Goldwasser-Micali

Génération des clés

- On choisit un entier n(modulo) tel que $n = pq$ avec p et q deux nombres premiers aléatoires assez grands ;
- On choisit $y \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ qui n'est pas un residu quadratique modulo n mais le sybole de jacobi est $(\frac{y}{n})_{jac} = 1$ (on dit que y est un pseudo-carre) $\frac{\mathbb{Z}}{n\mathbb{Z}}$;
- Clé publique (y, n) :
- Clé privée (p, q).

Goldwasser-Micali

Chiffrement

- On prend la clé publique (y, n) ;
- représenter le message m comme une chiane binaire de t bits. $m = m_1 m_2 m_3 \ldots . m_t$;
- Pour i allant de 1 à t, faire :
  - choisir $x \in \left(\frac{\mathbb{Z}}{n\mathbb{Z}}\right)^*$, aléatoire ;
  - si $m_i = 1$, on pose $c_i = yx^2 \ mod \ n$; si non on pose $c_i = x^2 \ mod \ n$;
- Texte chiffrée (la chine des c_i) : $C = (c_1, c_2, \ldots, c_t)$

Goldwasser-Micali

Déchiffrement

Pour déchiffrer un message chiffré $C \in \frac{\mathbb{Z}}{n\mathbb{Z}}$ on fait ce qui suit.

- Pour i allant de 1 à t faire,
  - On calcule le symbole de Jacobi $e_i = \left(\frac{c_i}{n}\right)_{jac}$
  - si $e_i = 1$, on pose $m_i = 0$; si non on pose $m_i = 1$;
- Message déchiffré : $m = m_1 m_2 \ldots . m_t$

Goldwasser-Micali

Preuve

Pourquoi ça marche le déchiffrement ?
$\Rightarrow$ Si $m_i = 0$ alors $c_i = x^2 mod n$ donc c_i est un redsidu quadratique d'où $e_i = 1$. mais seul le propriétaire de la clé privé (p, q) peut calculer le symbole de Jacobi *modulo* n de c_i car $\left(\frac{c_i}{n}\right)_{jac} = \left(\frac{c_i}{p}\right)_{leg}\left(\frac{c_i}{q}\right)_{leg}$.
$\Rightarrow$ Si $m_i = 1$, comme y est un pseudo-carré, alors $c_i = yx^2 mod n$ et donc c_i est un pseudo-carré. D'après une propriété sur les residus quadratique, c_i est residus quadratique *modulo* n ssi il l'est *modulo* p. Donc il suffit de calculer $\left(\frac{c_i}{p}\right)_{jac}$. Et comme seul le propriétaire des clés connait p, lui seul peut connaitre ce bit.

Goldwasser-Micali

Forces et faibblesses

On peut citer entre autres :

- Un des grands incovenients de cet algorithme est la taille du chiffre qui est constituer de $t \simeq log_2 m$ d'entiers de meme taille que n
- Son principale avantage est sa securite sementic car c'est un chiffrement bit par bit qui est sûr si la problème de la residuosité quadratique est difficile.

Merckle-Hellman

Merckle-Hellman

Sommaire

Merckle-Hellman

Présentation : Définition

Le " **problème du sac à dos**" connu en anglais sous le nom de "**Subset sum problème**" est le suivant ; soit $\{a_1, a_2, ..., a_n\}$ un ensemble d'entiers positifs non nuls (appelé "**Knapsack**") et un entier positif s, déterminer s'il existe ou non un sous ensemble des a_i dont la sommes est s c'est -à-dire s'il existe $x_i \in \{0, 1\}$ tels que $\sum_{1 \leq i \leq n} x_i a_i = s$

Cette version est un problème de décision qui est **NP-complet**. les meilleurs algorithmes de résolutions du "**problèmes du sac à dos**" tels que l'algorithme LLL sont exponentiels. Raison pour laquelle il est considéré comme difficile.

Merckle-Hellman

Présentation : Suite super croissante

Une suite $\{b_1, b_2, ..., b_n\}$ d'entiers positifs non nuls est super croissante si $b_i > \sum_{1 \leq j \leq i-1} b_j$ pour tout $1 < i \leq n$

Si on connait une suite super croissante alors on a une **instance facile** du "**problème du sac à dos**"

Merckle-Hellman

Algorithme de resolution du "probleme du sac à dos" simple c'est à dire définit par une suite super croissante.

Algorithme

Input : Une suite $\{b_1, b_2, ..., b_n\}$ super croissante et s un entier qui est une somme de ertains b_i.
Output : $\{x_1, x_2, ..., x_n\}$ avec $x_i \in \{0, 1\}$ tel que $\sum_{1 \leq i \leq n} x_i a_i = s$.

1. $i \leftarrow n$
2. Tant que $i > 1$ faire ce qui suit :
   - Si $s > b_i$ alors $x_i \leftarrow 1$ et $s \leftarrow s - b_i$, si non $x_i \leftarrow 0$;
   - $i \leftarrow i - 1$
3. Retourner $\{x_1, x_2, ..., x_n\}$

Merckle-Hellman

Algorithme de Merkle-Hellman : on prend une **"instance facile du probleme du sac à dos" en tant que suite super croissante** et on le masque en une suite en une suite aléatoire en utilisant une permutation aléatoire et un calcul modulo (qui ajoute de l'aleéa)

Algorithme de génération de clés

Bob fait ce qui suit :

1. Choisir une suite super croissante $(b_1, b_2, ..., b_n)$ et un module M tel que $M > b_1 + b_2 + ... + b_n$
2. Choisir un entier aleatoire W tel que $pgdc(W, M) = 1$
3. Choisir une permutation aléatoire π d'entiers de $\{1, 2, ..., n\}$;
4. Compute $a_i = Wb_{\pi(i)} mod\ M$, for $i = 1, 2, ..., n$
5. Clé publique de Bob $(a_1, a_2, ..., a_n)$; Clé privée de Bob $(\pi, M, W, (b_1, b_2, ..., b_n))$