"Introduction à la Sécurité Prouvée en Cryptographie à Clés publique"

Djiby SOW, Département de Mathématique et Informatique, Université Cheikh Anta Diop, Dakar

Ecole CIMPA - Université des Technologies et de Médecine- Nouakchott

22-23 Février 2016

Table des matières

Introduction

La principale mission de la cryptographie est de **garantir la sécurité des communications c'est-à-dire de permettre à des entités qui ne se font pas confiance en général de communiquer en toute sécurité en présence de potentiels adversaires (susceptibles entre autres d'accéder à des secrets en violant la confidentialité, d'intercepter et de modifier les informations échangées ou d'usurper des identités lors d'une communication)**.

Mais la cryptographie à elle seule ne prend pas en charge tous les besoins de sécurité et n'est donc pas la seule discipline qui intervienne dans la sécurité des communications. La cryptographie ne prend en charge **que les** 4 **premiers** sur les 5 services de sécurité fondamentaux que sont la **Confidentialité, l'intégrité, l'authentification, la Non-Répudiation et la Disponibilité**.

1. **Confidentialité :** Propriété qui assure que l'information n'est pas rendue disponible ou révélée à des personnes, entités ou processus non autorisés.
2. **Intégrité** : Propriété qui assure que des données n'ont pas été modifiées ou détruites de façon non autorisée lors de leur traitement, stockage et/ou transmission.
3. **Non répudiation / Signature numérique** : Fonction qui permet de garantir qu'aucun des partenaires d'une transaction ne pourra nier d'y avoir participé.
4. **Authentification** : Fonction permettant de prouver l'identité des partenaires (hommes, machines, réseaux,...) d'une communication ou l'origine des données lors d'un échange ou la validité d'un object cryptographique (clé, carte puce, certificat/signature,...).

Il existe d'autres services comme **la traçabilité, les preuves de connaissance, la contrôle d'accès, la certification, la gestion des secrets (clés) (qui pose un gros**

défis)

Sous l'angle des cryptosystèmes, la cryptographie est composés des systèmes à clés secrète et des systèmes à clés publique.

— **En cryptographie à clés secrète**, une même clés est utilisée pour chiffrer et pour déchiffrer ; en cas de deux clés, on s'assure que chacune d'elles, est facile à calculer à partir de l'autre (on parle aussi de cryptographie symétrique).

— **En cryptographie à clés publique**, on utilise deux clés dont l'une soit k' est difficile à déduire de l'autre soit k (on parle aussi de cryptographie non-symétrique). La clé k est appelée **clé publique** et est utilisée pour **le chiffrement ou la vérification de signature** selon le système ; la clé k' est appelée **clé privée** et est utilisée pour **le déchiffrement ou la signature** selon le système.

En cryptographie, **il était coutume de dire que tant qu'un système n'est pas attaqué, on peut continuer de l'utiliser**. Mais, le fait que des attaques n'aient pas réussi ou que des faiblesses n'aient pas été encore décelées sur un système cryptographique, ne garantit pas quelles ne le seront pas c'est à dire ne signifie pas que le système est sûr.

Un exemple classique est le cryptosystème de Chor-Rivest crée en 1985 et basé sur le "problème du sac à dos" qui résista pendant 15 ans à toute attaque avant qu'il ne soit complètement cassé par Serge Vandenay en 1998 : Crypto98.

La sécurité d'un système de cryptographie doit alors être étudiée au delà des attaques réelles connues. Mais comment ?

Que sécuriser dans un système de cryptographie ? Qu'est-ce-qu'on ne veut pas que l'attaquant soit capable de faire ? C'est la réponse à cette question qui permet de construire un modèle mathématique cohérent.

De façon formelle, **la sécurité prouvée se veut une démarche rigoureuse qui permet de montrer qu'une notion de sécurité est atteinte dans un système de cryptographie (tant qu'un certain problème mathématique réputé difficile n'est pas résolu par exemple pour la cryptographie à clés publique !)**.

Une **notion de sécurité** est un couple composé **d'un objectif de sécurité et d'un modèle d'attaquant**.

— L'objectif de sécurité spécifie ce qu'on **l'on souhaite concrètement protéger dans**

un cryptosystème (par exemple pour la cryptographie à clés publique, on peut citer les objectifs suivants : **Incassabilité, Fonction à sens unique, Indistinguabilité et Non-Malléabilité**).

— Le modèle de l'attaquant spécifie **les moyens et la puissance de calcul** supposés être à la disposition de l'attaquant pour tenter le calcul que nécessite son attaque.

L'efficacité d'une attaque contre l'atteinte d'une notion de sécurité est exprimée en terme d'avantage de l'attaquant, qui évalue quand-est-ce que l'attaquant a intérêt à tenter une attaque ? **L'avantage de l'attaquant** $\mathrm{Adv}(A) = |P(S_1) - P(S_0)|$ calcule la différence entre

— la probabilité $P(\mathcal{S}_1)$ d'un scénarios $\mathcal{S}_1$ où l'attaquent met en ?uvre tous les moyens à sa disposition pour réussir (connaissance du challenge, accès à des oracles de chiffrement et de déchiffrement,...),

— la probabilité $P(\mathcal{S}_0)$ d'un scénarios $\mathcal{S}_0$ où l'attaquant ou bien un simulateur, agis avec un maximum de vraisemblance (n'utilise pas les outils qu'il pourrait avoir à sa disposition et donc en répond au hasard).

Enfin avec la notion d'avantage on définit la notion de **niveau d'insécurité** d'un cryptosystème CS relativement à une notion de sécurité NS (dont l'ensemble des potentiels attaquant est Att(CS)) par

$$\mathrm{Insec}^{\mathrm{NS}}(\mathrm{CS}) = \max_{A \in \mathrm{Att}(\mathrm{CS})} \mathrm{Adv}(A)$$

Il est nécessaire de pouvoir identifier les paramètres du systèmes liés à la notion de sécurité ciblée et de pouvoir choisir la taille des paramètres en question de sorte que le niveau d'insécurité soit négligeable.

Chapitre 1

Notions de complexité

1.1 Généralités

1.1.1 Concepts de complexité

Définition 1.1.1. *L'objet de la théorie de la complexité est :*

- *pour les* ***algorithmes****, d'évaluer le nombre d'opérations élémentaires (****complexité temporelle****) et l'espace mémoire (****complexité spatiale****) nécessaire pour leur exécution ;*
- *pour les* ***problèmes (de décision)****, de les classifier suivants leur niveau de difficulté.*

⇒ Pour la complexité temporelle, cette classification doit mesurer la difficulté intrinsèque du problème via un algorithme bien détermine, donc elle ne doit dépendre ni de la classe de modèles de calcul, ni des ressources matérielles mobilisées (nombre de machines,...).

⇒ La principale préoccupation, c'est **le temps de calcul en fonction de la taille des donnes d'entre dans les algorithmes** car le problème de l'espace mémoire se pose de moins en moins avec le développement de la technologie.

Ainsi, on pourra définir **la difficulté calculatoire d'un problème relativement la complexité de l'algorithme le plus rapide parmi ceux qui le résolvent**.

Temps d'exécution

⇒ On admet que pour évaluer le temps d'exécution (= coût ou complexité temporelle)

d'un algorithme sur une donne d'entre de taille n, il suffit d'évaluer le nombre d'opérations élémentaires. Pour définir une opération élémentaire, on prend comme référence l'ordinateur théorique le plus simple qu'on appelle **machine de Turing** (dont l'exposé sort du cadre de ce cours). Pour simplifier, les opérations suivantes sont considérés comme élémentaires :

— les opérations booléennes,

— les affectations ;

— les instructions de contrôles,

— les comparaisons ;

— ...

Taille des données en entrée

Pour la taille des donnes d'entrées, on considère les cas suivants :

— la taille des nombres (nombre de symboles nécessaires pour l'écriture dans une base fixée) ;

— la taille des tableaux ;

— la taille des listes ;

— ....

Exemple :

— La taille d'un entier m (en binaire) est $n = |m| = E(\frac{\ln m}{\ln 2}) + 1$ où E est la partie entière ;

— La taille d'un polynôme de degré k avec des coefficients de taille n est $n(k+1)$;

— La taille d'une matrice avec a lignes et b colonnes avec des coefficients de taille n est abn.

Types de complexités

Notation grand tau : Soient f et g deux fonctions positives à variable entière. On écrit $f(n) = \mathcal{O}(g(n))$, s'il existe une constante $c > 0$ et un entier n_0 tels que $\forall n \geq n_0$, $0 \leq f(n) \leq cg(n)$. On dit que g domine f ou que f ne croit pas plus vite que g multiplié par une constante.

Dans le calcul de la complexité, on cherche simplement l'ordre de grandeur du coût temporel relativement à la taille des données d'entrée. Donc, on utilise

essentiellement la notation grand tau $\mathcal{O}$; si n est la taille des données d'entrée et $C(n)$ le coût temporel associé pour un algorithme bien déterminé on veut écrire $C(n) = \mathcal{O}(h(n))$, où h est une fonction (positive à variable entière) comparée avec fonctions classiques comme :

— les fonctions logarithmes $(\ln n)^\beta,\ \beta > 0$

— les fonctions polynômes ou puissances $n^\alpha,\ \alpha > 0$,

— les fonctions exponentielles $a^n,\ a > 1$

Croissance comparée

$$\lim_{n\mapsto\infty} \frac{n^\alpha}{a^n} = 0 \text{ et } \lim_{n\mapsto\infty} \frac{(\ln n)^\beta}{n^\alpha} = 0$$

Calcul complexité :

On note d les donnes de taille $\leq n$, $C(d)$ le coût associé l'exécution de la donne d par un algorithme et $C(n)$ la complexité de l'algorithme. alors on peut définir $C(n)$ de trois façons différentes ;

1. **Complexité dans le pire des cas** : $C(n) = \max\{C(d), d = \text{donnée de taille} \leq n\}$
2. **Complexité en moyenne** : $C(n) = \sum_{d=\text{donnée de taille}\leq n} P(d)C(d)$ où $P(d)$ est la probabilité d'exécuter la donnée d
3. **Complexité dans le meilleur des cas** : $C(n) = \min\{C(d), d = \text{donne de taille} \leq n\}$

Pour évaluer la complexité on considère toujours celle dans le **pire des cas** car on est seulement intéressée par les ordres de grandeur.

Définition 1.1.2. .

1. *Un algorithme est (à **complexité**) **polynômiale** si sa complexité dans le pire des cas est de la forme $\mathcal{O}(n^k)$ où $k > 0$ est une constante.*
2. *Si la complexité dans le pire des cas est de la forme $\mathcal{O}(a^n)$ où $a > 1$ est une constante, on dit que l'algorithme est (à **complexité**) **exponentielle**.*
3. *Cas intermédiaire : Un algorithme est (à complexité) **sous-exponentielle** si sa complexité est une fonction de la forme $L_q[\alpha, c] = \mathcal{O}\big(\exp\big((c+\mathcal{O}(1))(\ln q)^\alpha(\ln\ln q)^{1-\alpha}\big)\big)$ où $c > 0$ et $0 < \alpha < 1$: on dit que cette formule mesure la complexité sous-exponentielle, car sa complexité est :*

   — *polynomiale en $\ln q$ pour $\alpha = 0$: $L_q[0, c] = \mathcal{O}\big((\ln q)^{c+O(1)}\big)$*

— *exponentielle en* $\ln q$ *(car polynomiale en* q*) pour* $\alpha = 1$ *:* $L_q[1, c] = \mathcal{O}\big(q^{c+\mathcal{O}(1)}\big)$

Coût de complexité :

Notation	**Classification**
$C(n) = \mathcal{O}(1)$	Coût constante
$C(n) = \mathcal{O}(\ln n)$	Coût logarithmique
$C(n) = \mathcal{O}(n)$	Coût linaire
$C(n) = \mathcal{O}(n \ln n)$	Coût quasi-linaire
$C(n) = \mathcal{O}(n^2)$	Coût quadratique
$C(n) = \mathcal{O}(n^p)$	Coût polynomiale
$C(n) = \mathcal{O}(n^{\ln n})$	Coût quasi-polynomiale
$C(n) = \mathcal{O}(a^n)$	Coût exponentielle
$C(n) = \mathcal{O}(n!)$	Coût factorielle

Étude comparative

⇒ On suppose qu'on a un ordinateur qui exécute 10^9 instructions par secondes. On suppose aussi que les algorithmes utilisent au plus 1000 instructions par opération élémentaire. Donc, si on implémente l'algorithme sur la machine, il aura une vitesse de 10^6 opérations élémentaires par seconde.

⇒ On suppose que si un algorithme a pour complexité $O(m)$ alors il effectue au plus m opérations élémentaires (en négligeant la constante).

Ainsi, le coût de la complexité en secondes (s) ou microsecondes (μs) est : $C = \frac{m}{10^6}\ s = m\ \mu s$.

D'où le tableau suivant :

	$n = 10$	$n = 100$	$n = 1000$
$\ln n$			
$\sqrt{n}$	$3 \times 10^{-2} s$	$10^{-2} s$	$3 \times 10^{-2} s$
n	$10^{-5} s$	$10^{-4} s$	$10^{-3} s$
$n \ln n$	$2,3 \times 10^{-5} s$	$4,6 \times 10^{-4} s$	$6,9 \times 10^{-3} s$
n^2	$10^{-4} s$	$10^{-2} s$	1
n^3	$10^{-3} s$	$1s$	17minutes
n^5	$10^{-1} s$	$3h$	32ans
2^n	$1,02 \times 10^{-3} s$	4×10^{15} siecle	Astronomique
$n!$	$3,6s$	Astrnomique	Astronomique

1.2 Classification et Problèmes de décision

1.2.1 Classification

Définition 1.2.1. *Un problème de décision est un problème dont la réponse est OUI ou NON.*

Définition 1.2.2. .

1. ***La classe de complexité*** **P** *est l'ensemble des problèmes de décision résoluble en temps polynomiale.*
2. ***La classe de complexité*** **NP** *est l'ensemble des problèmes de décision pour lesquels la réponse OUI peut être vérifie en temps polynomiale moyennant une information supplémentaire appelle "certificat".*
3. ***La classe de complexité*** $\mathbf{Co-NP}$ *est l'ensemble des problèmes de décision pour lesquels la réponse NON peut être vérifie en temps polynomiale moyennant un "certificat" approprié.*

Remarque : Pour le problème du certificat, la définition **NP** (resp : $\mathbf{Co-NP}$) ne stipule pas que le certificat peut être trouvé en temps polynomiale mais que **s'il existe et s'il est connu, alors il peut être utilisé en temps polynomiale pour vérifier qu'on a OUI (resp : NON)**.

Exemple de problème NP

Problème : *"Composé"*

Instance de *"Composé"* : Un entier n ;

Question : Est que n est composé ? c'est dire existe-t-il deux entiers $a > 1$ et $b > 1$ tels que $n = ab$.

Nature du problème *"Composé"* : le problème *"Composé"* est $\mathbf{NP}$ (resp : $\mathbf{Co-NP}$) car, on peut vérifier en temps polynomiale si n est composé ou pas, à condition de disposer d'un diviseur (qui joue la rôle de certificat).

Problèmes ouverts :

On sait que $\mathbf{P} \subseteq \mathbf{NP}$ et $\mathbf{P} \subseteq \mathbf{Co-NP}$.

Mais, à notre connaissance les question suivantes sont ouvertes : $\mathbf{P} = \mathbf{NP}$?, $\mathbf{NP} = \mathbf{Co-NP}$? et $\mathbf{P} = \mathbf{NP} \bigcap \mathbf{Co-NP}$?

1.2.2 Principe de la réduction

Définition 1.2.3. ***Modèle***

Soient $\mathbb{D}_1$ et $\mathbb{D}_2$ deux problèmes de décision. On dit que $\mathbb{D}_1$ est **polynomialement réductible** *$\mathbb{D}_2$, (not : $\mathbb{D}_1 \preceq_{\mathbf{P}} \mathbb{D}_2$) s'il existe un algorithme $\mathcal{B}$ utilisant éventuellement un algorithme $\mathcal{A}$ qui résout $\mathbb{D}_2$ et vérifiant les deux conditions suivantes :*

- *le nombre d'appels de $\mathcal{B}$ à $\mathcal{A}$ est majoré par une fonction polynomiale de la donnée d'entrée ;*
- *le coût de $\mathcal{B}$ (hors appels $\mathcal{A}$) est polynomial ;*

Ainsi, si $\mathcal{A}$ résout $\mathbb{D}_2$ en temps polynomial alors $\mathcal{B}$ résout $\mathbb{D}_1$ en temps polynomial.

On interprète, $\mathbb{D}_1 \preceq_{\mathbf{P}} \mathbb{D}_2$ en disant que $\mathbb{D}_2$ est aussi difficile que $\mathbb{D}_1$ où que $\mathbb{D}_1$ n'est pas plus dur que $\mathbb{D}_2$.

Définition 1.2.4. ***Propriétés***

Soient $\mathbb{D}_1$, $\mathbb{D}_2$ et $\mathbb{D}_3$ trois problèmes de décision.

*i) Si $\mathbb{D}_1 \preceq_{\mathbf{P}} \mathbb{D}_2$ et $\mathbb{D}_2 \preceq_{\mathbf{P}} \mathbb{D}_3$ alors $\mathbb{D}_1 \preceq_{\mathbf{P}} \mathbb{D}_3$ (**Transitivité**)*

ii) Si $\mathbb{D}_1 \preceq_{\mathbf{P}} \mathbb{D}_2$ et $\mathbb{D}_2 \in \mathbf{P}$, alors $\mathbb{D}_1 \in \mathbf{P}$.

Équivalence

On dit que deux problèmes $\mathbb{D}_1$ et $\mathbb{D}_2$ de décision sont *polynomialement équivalents* s'ils sont (mutuellement) polynomialement réductibles l'un l'autre c'est dire : $\mathbb{D}_1 \preceq_{\mathbf{P}} \mathbb{D}_2$ et $\mathbb{D}_2 \preceq_{\mathbf{P}} \mathbb{D}_1$

1.2.3 Problème NP-Complet, Problème NP-Dur

Définition 1.2.5. ***Problème NP-Complet***

Un problème de décision est dit **NP***-complet si :*

i) $\mathbb{D} \in \mathbf{NP}$ *cela veut dire qu'il est possible de vérifier une solution efficacement (en temps polynomial)*

ii) $\mathbb{D}_1 \preceq_{\mathbf{P}} \mathbb{D}$ *pour tout* $\mathbb{D}_1 \in \mathbf{NP}$ *cela veut die que tous les problèmes de la classe* **NP** *se ramènent à celui-ci via une réduction polynomiale.*

Les problèmes **NP**-complets sont les problèmes les plus difficiles dans **NP** c'est dire qu'ils sont aussi difficiles que tout autre problème de **NP**

Définition 1.2.6. ***Problème NP-Dur***

Un problème $\mathbb{D}$ *est* **NP-*Dur*** *s'il existe un problème* $\mathbb{H}$ *qui est* **NP-*Complet*** *et qui se réduit* polynomialement $\mathbb{D}$ *c'est dire* $\mathbb{H} \preceq_{\mathbf{P}} \mathbb{D}$.

Un problème **NP-Dur** n'est pas forcément un problème de décision mais il est plus difficile qu'un problème **NP-Complet**. D'ailleurs tout problème **NP-Complet** est **NP-Dur**.

Nota bene : Pour connaitre les Problèmes **NP**-complet lire

— le live de Garey et Johnson :"Computers and Intractability : A Guide to the Theory of NP-Completeness" New York : W. H. Freeman. 1983. ISBN 0-7167-1045-5 ;

— la liste de Paul E. Dunne. An Annotated List of Selected NP-complete Problems. Université de Liverpool, Département d'informatique, COMP202, available online

Chapitre 2

Problème difficiles, Fonctions à sens unique et Fonction de hachage

2.1 Problème difficiles

Un problème est considéré comme **difficile** s'il est prouvé NP-dur ou s'il admet une large classe d'instances dont l'algorithme le plus rapide, connu pour résoudre ces instances, est exponentiel.

NB : Si la complexité d'un problème est sous exponentiel (comme la factorisation par exemple), on peut toujours choisir les paramètres pour que le problème reste difficile relativement à la puissance de calcul actuel.

Quatre problèmes fondamentaux et pratiques :

— Problème du logarithme discret ;
— Problème de la factorisation ;
— Problème de la recherche du plus court vecteur dans un réseau (SVP : chort vector problem) ;
— Problème de la recherche du vecteur le plus proche d'un vecteur donné dans un réseau (CVP : closed vector problem) ;

2.2 Problème du Logarithme discret

Présentation du problème du Log discret

Enoncé 2.2.1. ***Problèmes du Logarithme discret : PLD :***

Soit G un groupe (noté multiplicative) fini et H un sous groupe cyclique d'ordre premier.

Soit g un générateur de H. On note $n = ord(g) = \#H$ l'ordre de g.

"Pour $y \in H$ tiré au hasard, trouver $0 \leq x < n$ tel que $y = g^x$ dans G".

Résolution

Il existe plusieurs algorithmes (avec plusieurs améliorations pour chacun d'eux) pour résoudre le logarithme discret. Les principaux sont :

— Algorithme "Pas de Bébé-Pas de Géant" ;

— Algorithme "Pollard's Rho"

— Algorithme "Pohling-Hellman" ;

— Algorithme "Index-Calculus".

Il est communément admis que PLD est difficile dans certains groupes : les meilleurs groupes connus sont sur les courbes elliptiques même s'il existe existe des attaques efficaces récentes (Joux, Menezes et *al.*) sur les courbes elliptiques sur les corps de petite caractéristique (à travers les couplages).

2.2.1 Algorithme Pas de Bébé-Pas de Géant pour le Log discret

Position du problème

$\Rightarrow$ Si $y = g^x$, on peut faire une recherche exhaustive en calculant g^i pour i allant de 1 $n = ordg$. Ce qui est très couteux.

$\Rightarrow$ L'algorithme **"Bebe step-geant step"** offre une meilleur alternative avec un **compromis temps-mémoire** pour calculer le logarithme discret

$\Rightarrow$ On peut fixer un entier m et faire la division euclidienne avec x pour avoir $x = im + j$ d'o $y = g^{im+j} = (g^m)^i g^j$ alors $y(g^{-m})^i = g^j$.

$\Rightarrow$ On calcule les valeurs g^j qu'on garde dans un tableau puis on calcule progressivement les $y(g^{-m})^i$ et on cherche une collusion avec les éléments du tableau.

$\Rightarrow$ Dans $g^{im+j} = (g^m)^i g^j$, l'entier m est le pas de Géant et l'entier i est le nombre de pas de Géant, alors que 1 est le pas de Bébé et l'entier j est le nombre de pas de Bébé.

Algorithme 2.2.1. ***Algorithme "Bebe step-geant step"***

Input : g un générateur d'ordre n d'un groupe cyclique G de cardinal n et y un élément de G.

Output : le logarithme discret $x = Log_g y$.

- $m \leftarrow E(\sqrt{n})$
- *Calculer les g^j pour j allant de 0 m et sauvegarder les couples (j, g^j) dans un tableau en rangeant les g^j par ordre croissante (voir algorithmes de Tri)*
- *Calculer g^{-m} et $\gamma \leftarrow y$*
- *Pour i allant de 0 $m-1$, faire :*
  - *Tester si γ est dans le tableau de sauvegarde ;*
  - *Si $\gamma = g^j$: Retourner $x = im + j$*
  - *Si non : $\gamma \leftarrow \gamma g^{-m}$*

Complexité

$\Rightarrow$ 1) le calcul de m se fait en coût constant ;

$\Rightarrow$ 2) le calcul de g^j se fait en $O(\ln j)$ et comme le stockage d'un élément se fait en coût constant et qu'il y a $\sqrt{n}$ éléments stocker, alors la complexité pour (1) et (2) est de $O(\sqrt{n} + \sum_{j \leq \sqrt{n}} \ln j) = O(\sqrt{n} + \ln \sqrt{n}!) = O(\sqrt{n} \ln \sqrt{n}) = O(\sqrt{n} \ln n)$.

$\Rightarrow$ 3) Pour trier les éléments du tableau, il faut $O(\sqrt{n} \ln \sqrt{n}) = O(\sqrt{n} \ln n)$ comparaisons en utilisant le Tri-Fusion.

$\Rightarrow$ 4) Le ème point est en $O(\sqrt{n})$ multiplications et $O(\sqrt{n})$ comparaisons car le nombre de pas est borné par $\sqrt{n}$ alors que l'affection est en coût constant donc il ne reste que la multiplication prendre en compte.

$\Rightarrow$ En conclusion, si on considère qu'une multiplication coût plus de $\ln n$ comparaisons alors la complexité total est de $O(\sqrt{n})$ opérations dans G ; Pour le coût réel, il faut connaitre le coût d'une multiplication dans le groupe considéré.

2.2.2 Algorithme Rho Pollard pour le log discret

Preliminaire 1

Beaucoup d'algorithmes utilisent l'existence de collusions (coïncidences) $\big(x \neq x' \; et \; f(x) = f(x') \big)$ pour une fonction aléatoire $(f : E \to E)$.

Pour être sûr de l'existence d'une collusion, il suffit de définir récursivement une fonction f sur un ensemble fini E.

PÉRIODICITÉ Soit E un ensemble fini de cardinal n et $f : E \to E$ une fonction aléatoire . Soit $a \in E$, on définie une suite $(x_k)_k$ par $x_0 = a$ et $x_{k+1} = f(x_k)$. Alors la suite est ultimement périodique. Si x_{k_1} est le premier indice pour lequel il existe une collusion $x_{k_1} = x_{k_2}$ avec $k_2 < k_1$ alors en posant $T = k_1 - k_2$, on a $x_{k+T} = x_k$ pour tout $k \geq k_2$.

Nous allons introduire le paradoxe des anniversaires qui va permettre d'évaluer le nombre de termes nécessaire pour trouver une collusion (la première !) avec cette suite.

Paradoxe des anniversaires

Le calcul de la complexité de l'algorithme précédent nécessite de pouvoir évaluer la probabilité d'avoir une collusion en choisissant des valeurs dans un ensemble. Cette méthode est connu sous le nom de **"Paradoxe des anniversaires"**

Problème 1 Une classe compte k. Calculer la probabilité pour que 2 élèves fêtent leur anniversaire le même jour sachant que l'année compte 365 jours ?

Problème 2 : Cas général On tire au hasard k valeurs parmi n suivant une loi de probabilité uniforme. Calculer la probabilité pour qu'il y ait collusions (deux valeurs identiques) ?

On pose l'"univers" de l'épreuve comme tant

$\Omega =$ **ensemble des** k **listes de valeurs prises parmi** n.

Alors $Card\Omega = n^k$.

On pose :

$A=$ **l'événement : "**2 **valeurs parmi les** k **valeurs, coïncident"**.

Alors l'événement contraire est :

$\bar{A}=$ **"les** k **valeurs sont deux deux distincts"** ainsi,

$\bar{A}=$ **"ensemble des tirages successifs sans remise de** k **valeurs parmi** n**"**.

D'où la probabilité de $\bar{A}$ est :

$P(\bar{A}) = \frac{A_n^k}{n^k} = \frac{n!}{(n-k)!n^k} = \frac{n(n-1)....(n-k+1)}{n^k} = \prod_{i=1}^{k}(1-\frac{i}{n})$

Par suite $\mathbf{P(A) = 1 - \prod_{i=1}^{k}(1 - \frac{i}{n})}$

k=*nombre d'élèves* ; P= probabilité d'avoir 2 élèves qui fêtent leur anniversaire le même jour.

k	P
2	0,01
5	0,03
10	0,12
20	0,42

k	P
22	0,48
23	0,51
30	0,71
50	0,97

.

On voit que dès que $k \geq 23$ alors la probabilité d'avoir une collusion est supérieur $\frac{1}{2}$. Ainsi dans une classe de 23 personnes, on a une probabilité supérieur 1/2 d'y trouver deux élèves qui fêtent leur anniversaire le même jour.

Nous allons voir cette propriété plus en détaille.

Maintenant on va essayer d'évaluer pour quelle valeur de k, $P(A)$ est supérieur une valeur p_0 fixé, si n est connu.

On a $\mathbf{P(A) = 1 - \prod_{i=1}^{k}(1 - \frac{i}{n})}$.

or on sait que $1 - x \leq e^{-x}$, $\forall x \in \mathbb{R}_+$ donc en posant $x = \frac{i}{n}$ on a : $\prod_{i=1}^{k}(1-\frac{i}{n}) \leq \prod_{i=1}^{k} e^{\frac{-i}{n}} = e^{\sum_{i=1}^{k}\frac{-i}{n}} = e^{-\frac{k(k-1)}{2n}}$. D'o $\mathbf{1 - P(A) \leq e^{-\frac{k(k-1)}{2n}} \Leftrightarrow P(A) \geq 1 - e^{-\frac{k(k-1)}{2n}}}$. Par suite si on veut $P(A) \geq p_0$, il suffit de choisir k de sorte que $1 - e^{-\frac{k(k-1)}{2n}} \geq p_0$

Ainsi $\mathbf{1 - p_0 \geq e^{-\frac{k(k-1)}{2n}} \Leftrightarrow k^2 - k + 2n\ln(1 - p_0) \geq 0}$.

Or $\mathbf{k^2 - k + 2n\ln(1 - p_0) = 0}$ donne deux racines

$k_1 = \frac{1-\sqrt{1-8n\ln(1-p_0)}}{2} < 0$ et $k_2 = \frac{1+\sqrt{1-8n\ln(1-p_0)}}{2} > 0$

D'où, pour vérifier l'inégalité précédente, il faut $k \geq k_2 = \frac{1+\sqrt{1-8n\ln(1-p_0)}}{2}$.

Exercice : Utiliser cette formule pour vérifier que : pour avoir $p(A) \geq p_0 = \frac{1}{2}$, quand $n = 365$, il suffit que $k \geq 23$.

□

Maintenant par approximation, on a : $k \geq k_2 \cong \sqrt{-2n\ln(1-p_0)} = O(\sqrt{n})$.

Conclusion sur le paradoxe des anniversaires Si on tire aléatoirement des valeurs dans un ensemble de taille n, le nombre de tirage nécessaire pour qu'il y

ait collusion est de $\mathbf{O(\sqrt{n})}$

On rappelle qu'on a déjà vu que si E un ensemble fini de cardinal n et $f : E \to E$ une fonction aléatoire ; $a \in E$, la suite $(x_k)_k$ définis par $x_0 = a$ et $x_{k+1} = f(x_k)$ est ultimement périodique. De plus si x_{k_1} est le premier indice pour lequel il existe une collusion $x_{k_1} = x_{k_2}$ avec $k_2 < k_1$ alors en posant $T = k_1 - k_2$, on a $x_{k+T} = x_k$ pour tout $k \geq k_2$.

$\Rightarrow$ Pour trouver une collusion, on peut pratiquer la recherche exhaustive.

$\Rightarrow$ Pour cela, il faut partir d'une valeur x_0, calculer et stocker les $f(x_t)$ suivants et chaque pas, comparer la valeur nouvellement calcule celles dj stockes jusqu'à' ce qu'il y ait coïncidence (collusion).

$\Rightarrow$ Il faudra au moins stocker k_2 éléments et faire en moyenne $\frac{1}{2}.k_2(k_2 - 1)$ comparaisons. Avec le paradoxe des anniversaires, on estime que $k_2 = O(\sqrt{n})$ o $n = |E|$. Ainsi, la complexité spatiale est de $O(\sqrt{n})$ et la complexité temporelle est de $O(n)$. Et donc ces complexités sont exponentielles en la taille de n (en base 2 par exemple).

Pour baisser cette complexité, Knuth a décrit un algorithme appelé **Floyd's finding algorithm** qui ne coûte presque rien en mémoire et coûte peu en temps si la période n'est pas trop grande.

Le plus petit multiple de T qui soit supérieur k_2 est appel pacte de la suite et est note e.

Exemple 1 On prend $f(x) = x^2 + 1 \ mod \ 11$ et $x_0 = 1$. Vérifier que $k_2 = 3$ et $T = 2$ donc l'épacte est $e = 4$. Dessiner le schémas de la suite sous la forme d'un graphe.

Exemple 2 On prend $f(x) = x^2 + 1 \ mod \ 17$ et $x_0 = 3$. Vérifier que $k_2 = 1$ et $T = 6$ donc l'épacte est $e = 6$. Dessiner le schémas de la suite sous la forme d'un graphe.

Exemple 3 On prend $f(x) = x^2 + 1 \ mod \ 97$ et $x_0 = 13$. Déterminer k_2, T et e. Dessiner le schémas de la suite sous la forme d'un graphe.

Soit e l'épacte de la suite alors e est la première valeur telle que $x_{2e} = x_e$. or $x_{2e} = x_e \Leftrightarrow f^e(x_0) = (f \circ f)^e(x_0)$. Ainsi en partant de $f(x_0)$ et en comparant récursivement les $f(x_i)$ et $f^2(x_i)$, au bout de e tapes exactement, on trouve une collusion. Ceci constitue l'algorithme appel **Floyd's cycle finding algorithm**.

Algorithme 2.2.2. ***Input :*** *Un ensemble fini E de cardinal n, $f : E \to E$ une fonction et a un entier ;*

Output : *l'épacte e de la suite $(x_k)_k$ avec $x_0 = a$ et $x_{k+1} = f(x_k)$.*

1. *Initialiser :* $e \leftarrow 0$, $x \leftarrow a, y \leftarrow a$;
2. *Répéter*
   - $e \leftarrow 1$; $x \leftarrow f(x)$; $y \leftarrow f \circ f(y)$
   - *si* $x = y$ *retourner* e
3. *Fin répéter*

On considère G comme un groupe d'ordre premier de générateur g. On veut déterminer x le logarithme discret de $y = g^x$.

On divise G en trois parties de même taille G_1, G_2 et G_3. On définie la suite $(x_k)_k$ par $x_0 = 1$, $x_{k+1} = f(x_k) = \begin{cases} yx_k, & \text{Si } x_k \in G_1; \\ x_k^2, & \text{Si } x_k \in G_2; \\ gx_k, & \text{Si } x_k \in G_3. \end{cases}$

Alors, cette suite définie deux suites d'entiers $(a_k)_k$ et $(b_k)_k$ vérifiant $x_k = g^{a_k} y^{b_k}$ pour $k \geq 0$ et donnes par $a_0 = 0$, $b_0 = 0$, $a_{k+1} = \begin{cases} a_k, & \text{Si } x_k \in G_1; \\ 2a_k mod n, & \text{Si } x_k \in G_2; \\ a_k + 1 \ mod n, & \text{Si } x_k \in G_3. \end{cases}$

et

$b_{k+1} = \begin{cases} b_k + 1 \ mod n, & \text{Si } x_k \in G_1; \\ 2b_k \ mod n, & \text{Si } x_k \in G_2; \\ b_k, & \text{Si } x_k \in G_3. \end{cases}$

Le "Floyd's finding algorithm" déjà vu permet de trouver l'épacte e de la suite et alors on $x_{2e} = x_k \Leftrightarrow g^{a_{2e}} y^{b_{2e}} = g^{a_e} y^{b_e} \Leftrightarrow g^{a_{2e} - a_e} = y^{b_e - b_{2e}}$.

Mais, $y = g^x$ donc $x(b_e - b_{2e}) = a_{2e} - a_e \ mod n$.

Algorithme 2.2.3. ***Input :*** *Un groupe G de cardinal premier n et g un générateur de G et $y \in G$;*

Output : *x le log discret de y base g.*

1. *Initialiser :* $x_0 \leftarrow 1$, $a_0 \leftarrow 0, b_0 \leftarrow 0$;
2. *Pour $k = 1, 2, 3, ...,$ faire ce qui suit*

— *Avec les valeurs déjà calcules* x_{k-1}, a_{k-1}, b_{k-1} *et* x_{2k-2}, a_{2k-2}, b_{2k-2}, *calculer les nouvelles valeurs* x_k, a_k, b_k *et* x_{2k}, a_{2k}, b_{2k} *en utilisant les équations précédentes*

— *si* $x_k = x_{2k}$ *alors faire :* $r = b_k - b_{2k}\ modn$

si $r = 0$ *retourner échec*

si non calculer $r^{-1}\ modn$ *et retourner* $r^{-1}(a_{2e} - a_e)\ modn$

2.3 Problème de la Factorisation

Présentation du problème de la factorisation

Enoncé 2.3.1. ***La factorisation des entiers :***

Étant donné un entier positif $n = pq$, *produit de deux nombres premiers aléatoirement choisis, trouver* p *et* q.

Résolution : Il existe plusieurs algorithmes (avec plusieurs améliorations pour chacun d'eux) pour résoudre la factorisation. Les principaux sont :

— Algorithme "Rho Pollard " ;

— Algorithme "$p-1$ Pollard"

— Algorithme "Factorisation sur les courbes elliptiques" ;

— Number field sieve : quadratic sieve ou general number field sieve

Il est communément admis que la factorisation est difficile si p et q sont deux entiers premiers aléatoires de taille suffisamment grande.

2.3.1 Algorithme Rho Pollard pour la factorisation

Algorithme 2.3.1. ***Input :*** *Un entier composé* n *qui n'est la puissance d'un premier ;* ***Output :*** *Un facteur non trivial de* n *;*

1. *Initialiser :* $x \leftarrow a, y \leftarrow a$ *;*
2. *Pour* $i = 1, i = 2, ...$ *faire :*

   2.1 Calculer $x \leftarrow x^2 + b\ mod\ n$, $y \leftarrow y^2 + b\ mod\ n$, $b \leftarrow y^2 + b\ mod\ n$

   2.2 Calculer $d = gcd(x - y, n)$

   2.3 Si $1 < d < n$ *retourner* d, *afficher succès et terminer*

2.4 Si $d = n$, terminer en affichant échec ;

En fait, tous les algorithmes de factorisation cherchent construire des nombres x et y tels $x \neq +/-y$ et $x^2 = y^2 \ mod \ n$. Dans ce cas on obtient un diviseur $d = pgdc(x-y, n)$ si $d \neq n$.

$\Rightarrow$ Tout le problème c'est comment construire efficacement les nombres x et y de sorte que la probabilité d'avoir $d = n$ soit faible.

$\Rightarrow$ La méthode Rho (utilisé par d'autres algorithmes aussi tel que le logarithme discret), consiste chercher des collusions dans une suite du type $x_{k+1} = f(x_k)$ o $x_0 = a$ est appel "germe" et $f : E \to E$ est une **fonction aléatoire** d'un ensemble fini E.

$\Rightarrow$ Dans le cas de Rho Pollard, la fonction f est donne par $f(x) = x^2 + b \ mod \ n$ qui est généralement assez aléatoire et on cherche des collusions du type $x_{k'} = x_{2k'} \Leftrightarrow f^{k'}(x_0) = (f \circ f)^{k'}(x_0)$ c'est dire $f^{k'}(a) = (f \circ f)^{k'}(a)$.

Pour être simple dans l'algorithme Rho Pollard on prend $a = 2$ et $b = 0$ ou $b = 1$. Par exemple, si $d = n$, on peut reprendre en prenant un autre b.

$\Rightarrow$ L'algorithme de **Rho Pollard** est dérivé de celle de **Floyd's cycle finding algorithm** qui permet de chercher k' en se basant sur le fait que la suite (x_k) est ultimement périodique.

$\Rightarrow$ Le nombre de pas de l'algo de **Floyd's cycle finding algorithm** est estimé par le **"Paradoxe des anniversaires"** déjà vu.

2.3.2 Algorithme $p-1$ Pollard pour la factorisation

Soit B un entier positif, alors un entier n est dit B-friable (smooth) ou friable relativement B si tous les facteurs premiers de n sont inférieur B.

Soient n un entier multiple d'un nombre premier p, comment factoriser n pour trouver p ?

Si p est un nombre premier tel que $p-1$ est B-friable alors l'algorithme $p-1$ Pollard peut être utilisé pour trouver p de faon très efficace si B n'est pas trop grand (en pratique B est très petit devant n).

Soit Q le plus petit multiple commun des puissances de nombres premiers inférieur B et

n. Si $q^l < n$ alors $l \leq E(\frac{lnn}{lnq})$ o E est la partie entière.

Alors $Q = \prod_{q \leq B, \text{premier}} q^{E(\frac{lnn}{lnq})}$.

Maintenant, si a est un entier premier avec p alors d'après Fermat $a^{p-1} = 1 \ mod \ p$ or $p-1$ divise Q donc $a^Q = 1 \ mod \ p$. On calcule alors $d = pgcd(a^Q - 1, n)$ alors p divise d. Si $d \neq n$ on a un facteur non trivial de n en calculant $pgdc(d, n)$. Si $d = n$ on reprend.

Algorithme 2.3.2. ***Input :*** *Un entier composé n qui n'est la puissance d'un premier;* ***Output :*** *Un facteur non trivial de n ;*

1. *Choisir une borne friable B ;*
2. *Choisir un entier a, $2 < a < n-1$ et calculer $d = pgcd(a-1, n)$. Si $d > 2$ retourner d et terminer (si non continuer avec a) Pour tout entier premier $q < B$, faire*
   3.1 Calculer $l = E(\frac{lnn}{lnq})$
   3.2 Calculer $a \leftarrow a^{q^l} mod n$;
3. *Calculer $d = pgdc(a-1, n)$;*
4. *Si $d = 1$ ou $d = n$ retourner échec et terminer; si non retourner d et succès puis terminer.*

Complexité de l'algorithme $p-1$ Pollard

Soit n un entier premier ayant un facteur premier p qui est B-friable. Alors l'algorithme $p-1$ Pollard exécute $O(\frac{B \ ln \ n}{ln \ B})$ multiplications modulo n.

Donc si $p-1$ (et donc B) est divisible par un grand nombre premier, l'algorithme $p-1$ Pollard ne sera pas efficace car B sera trop grand et il y aura alors trop de nombres premiers inférieur B qu'il faut calculer et utiliser dans le corps de l'algorithme.

2.4 Problèmes SVP et CVP dans les Réseaux arithmétiques

Soient n et d deux entiers positifs. Soit L une partie non vide de $\mathbb{R}^n$. On dit que L est un réseau s'il existe une famille libre $(b_1, b_2, ..., b_d)$ de $\mathbb{R}^n$ telle que

$$L = \sum_{i=1}^{d} \mathbb{Z} b_i = \left\{ \sum_{i=1}^{d} x_i b_i, x_i \in \mathbb{Z} \right\}$$

L'entier d est la dimension du réseau, et $(b_1, b_2, ..., b_d)$ est une base de ce réseau.

Un problème très important en théorie des nombres et en cryptanalyse est la recherche d'un vecteur dans un réseau. Deux sous-problèmes se posent alors.

$\Rightarrow$ **The Shortest Vector Problem (SVP))** :

Étant donné un réseau L, déterminer un vecteur non nul v qui minimise la norme $\| v \|$.

$\Rightarrow$ **The Closed Vector Problem (CVP))** :

Étant donne un réseau L et un vecteur v_0, déterminer un vecteur $v \neq v_0$ qui minimise la norme $\| v - v_0 \|$.

$\Rightarrow$ **Résolution** :

L'algorithme LLL (Lenstra, Lenstra and Lovasz 1982) résout le SVP. Il est communément admis que la SVPt e CVP sont des problèmes difficiles si le réseau est bien choisi, de taille suffisamment grande.

En 2005, Regev a introduit le LWE sur les réseaux.

2.5 Fonctions à sens unique (One way function)

Définition 2.5.1. — *Une fonction $f : A \longrightarrow B$ est dite à sens unique s'il est facile à calculer $f(x)$ pour tout $x \in A$ (complexité polynômiale), il est difficile (complexité exponentielle) pour presque tout $y \in B$, de trouver x tel que $y = f(x)$.*

— *Une fonction est à sens unique avec trappe si l'on connaît un secret permettant de l'inverser.*

Exemples

1. **Groupes**

   $R : [1, n-1] \longrightarrow G : x \mapsto g^x$ (g un générateur d'un sous groupe cyclique H de G, $n = \#H$).

2. **Arithmétique modulaire sur les entiers**

   $T : (\mathbb{Z}/n\mathbb{Z})^* \longrightarrow (\mathbb{Z}/n\mathbb{Z})^* : x \mapsto x^e$ où $n = pq$ produit de deux nombres premiers très grands et $e \wedge \varphi(n) = 1$, e est impair.

3. **Réseaux arithmétiques : Arithmétique modulaire sur les polynômes**

   Soient p et q deux entiers $p << q$. On note $R_q = \frac{\mathbb{Z}/q\mathbb{Z}[X]}{(X^n-1)}$, $R_p = \frac{\mathbb{Z}/p\mathbb{Z}[X]}{(X^n-1)}$

   $L : R_p \times R_p \longrightarrow R_q : (u, v) \longmapsto hu + v$ ($h \in R_q \subset R_p$, est fixé et non inversible)

Applications : Les fonctions à sens unique sont essentiellement utilisées pour construire des systèmes cryptographiques.

2.6 Fonctions de hachage

$\{0,1\}^\star$ ensemble des chaines de longueur quelconque, $\{0,1\}^l$ ensemble des chaines longueur fixe $l \in \mathbb{N} \setminus \{0\}$,

Définition 2.6.1. *Une fonction $H : \{0,1\}^\star \longrightarrow \{0,1\}^l$ est dite fonction de hachage si :*

1. *Pour tout x, $H(x)$ est facile à calculer, $H(x)$ est appelé le* ***hash ou l'empreinte*** *de x*
2. *Étant donné $H(x)$, il est difficile de trouver y tel que $y = H(x)$ (fonction à sens unique) ;*
3. *Étant donné x, il est difficile de trouver x' tel que $x \neq x'$ et $H(x) = H(x')$, (collusions faibles) ;*
4. *Il est difficile de trouver x et x' (de son choix) $x \neq x'$ tel que $H(x) = H(x')$, (collusions fortes) ;*

Exemples MD5, SHA1, SHA2, SHA3 (conception en cours : visiter sha3.org) sont des fonctions de hachage normalisées au cours du temps.

Applications :

Les fonctions de hachage ont de nombreuses applications :

— L'authentification de messages :

— L'optimisation des schémas de signature :

— La confirmation sans divulgation de connaissance :

— L'identification basée sur les mots de passe :

— La construction de générateurs pseudo-aléatoires :

Chapitre 3

Modélisation des cryptosystèmes à clés publique

3.1 Chiffrement à clés publique

3.1.1 Modélisation du chiffrement à clés publique

Un système de cryptographie est composé d'un quintuplet $(\mathcal{P}, \mathcal{C}, C_k, D_{k'}, \mathcal{K})$ où :

1. $\mathcal{P}$ est un ensemble appelé espace des textes clairs
2. $\mathcal{C}$ est un ensemble appelé espace des textes chiffrés
3. $\mathcal{K}$ est un ensemble appelé espace des clés
4. $Gen_{\mathcal{K}}$ un algorithme de génération de clés (=les éléments de $\mathcal{K}$) ;
5. **Cas déterministe**
   - $C_k : \mathcal{P} \to \mathcal{C}$ est une fonction à sens unique (avec trappe qui est un inverse à gauche) appelée fonction de chiffrement et qui dépend d'un paramètre k appelé clé.
   - $D_{k'} : \mathcal{C} \to \mathcal{P}$ est la trappe (la fonction inverse gauche de C_k i.e $D_{k'} \circ C_k(m) = m, \forall m \in \mathcal{P}$) et est appelée fonction de déchiffrement (dépendant de la clé k'.)
6. **Cas probabiliste**
7. $\mathcal{R}$ est un ensemble appelé espace des nonces (valeurs permettant de probabiliser le système)

8. $C_k : \mathcal{P} \times \mathcal{R} \to \mathcal{C}$ est une fonction à sens unique (avec trappe qui est un inverse à gauche) appelée fonction de chiffrement et qui dépend d'un paramètre k appelé clé.
9. $D_{k'} : \mathcal{C} \to \mathcal{P}$ est la trappe (la fonction inverse gauche de C_k i.e $D_{k'} \circ C_k((m,r)) = m, \forall (m;r) \in \mathcal{P} \times \mathcal{R}$) et est appelée fonction de déchiffrement (dépendant de la clé k'.)

Comme C_k est à sens unique alors nécessairement k' est difficile à calculer à partir de k. Donc, on peut publier k sans danger et on garde k' privée. On dit qu'on est en cryptographie à clé publique. Chaque partenaire a sa clé publique/privée. Alice $(P_{k,A}, S_{k,A})$, Bob $(P_{k,B}, S_{k,B})$.

L'expéditeur utilise la clé publique du destinataire.

La cryptographie à clé publique est basée sur les fonctions à sens unique avec trappe. Les fonctions à sens unique sont basées sur des problèmes mathématiques difficiles.

NB : Pour la cryptographie à clés sécrète une utilise une seule clés pour le chiffrement te le déchiffrement ou chacune des clés k et k' est calculatoirement facile à déduire de l'autre ; raisons pour laquelle on l'appelle aussi cryptographie symétrique.

3.1.2 Exemple 1 : Chiffrement El Gamal 1976

Génération des clés Alice choisit :

— G un non groupe (c'est à dire le DLP est difficile) avec $H \leq G$ un sous-groupe cyclique de G, , $\#H = q$;

— g un générateur de H et $a < q$ un entier aléatoire (à choisir correctement) ;

— clé publique d'Alice (g, g^a, G) et clé privée d'Alice a.

Chiffrement Bob :

— prend la clé publique d'Alice (g, g^a, G) ;

— choisit un message $m \in H$;

— génère une valeur aléatoire $k \leq q$;

— calcule $c_1 = g^k$ et $c_2 = m(g^a)^k$;

— chiffré $c = (c_1, c_2)$.

Déchiffrement Alice prend sa clé privée a puis calcule $c_1^{q-a} \cdot c_2 = (g^k)^{q-a} m g^{ka} = m$ car $g^{q-a} = g^{-a}$

NB : Si le logarithme discret est facile, on résout le chiffrement El Gamal, mais la réciproque est fausse.

On va montrer plus loin que le chiffrement El Gamal est équivalente au problème calculatoire de Diffie-Hellman qui est :

CDH "étant donné g^a et g^b où a et b sont deux valeurs aléatoires inférieures ou égales à q et inconnues, calculer g^{ab}".

On sait que DLP $\Longrightarrow$ CDH, mais en utilisant ce qu'on appelle couplage sur les courbes elliptiques, on peut montrer que CDH $\not\Longrightarrow$ DLP.

3.1.3 Exemple 2 : Chiffrement RSA (Rivest, Shamir et Adleman 1978)

Génération des clés : Alice

— génère p et q deux nombres premiers $|p| = |q|$ suffisamment grand ;

— calcule $\varphi(n) = (p-1)(q-1)$;

— Choisit e tel que $e \wedge \varphi(n) = 1$ et calcule d tel que que $ed = 1 \mod \varphi(n)$;

— clé publique (e, n), clé privée d, $\varphi(n)$ est à écraser.

Chiffrement : Bob :

— prend la clé publique d'Alice (e, n) ;

— choisit un message $m \in \mathbb{Z}/n\mathbb{Z}$;

— calcule $c = m^e \mod n$ qui est le chiffré.

Déchiffrement Alice prend sa clé privée d puis calcule $c^d \mod n = m^{ed}$, comme $\exists k$ tel que $ed - k\varphi(n) = 1$ alors $m^{ed} = m^{1+k\varphi(n)} = m.(m^{\varphi(n)})^k$. Si m est inversible alors $m^{\varphi(n)} = 1$ et donc $c^d \mod n = m$.

NB : Noter que la probabilité pour que m ne soit pas inversible est $\frac{\varphi(n)}{n} = (1-\frac{1}{p})(1-\frac{1}{q})$, ce qui est presque égale à 1. De plus si m n'est pas inversible alors on trouve un facteur de n.

3.1.4 Exemple 3 : Chiffrement NTRU (Hoffstein, Pipher et Silverman 1998)

Génération des clés : Alice

— choisit p et q deux entiers $p << q$, pose $R_q = \frac{\mathbb{Z}/q\mathbb{Z}[X]}{(X^n-1)}$, $R_p = \frac{\mathbb{Z}/p\mathbb{Z}[X]}{(X^n-1)}$

— choisit un polynôme $f \in R_p$ inversible modulo p ($f_p = f^{-1} \mod p$)) et modulo q ($f_q = f^{-1} \mod q$)

— Choisit aléatoirement un polynôme $g \in R_p$ et calcule $h = g \times f_q \mod q$;

— clé publique (R_p, R_q, h), clé privée $(f_p = f^{-1} \mod p, f)$, g est à écraser.

Chiffrement : Bob

— prend la clé publique d'Alice (R_p, R_q, h) ;

— choisit un message $m \in R_p$;

— Choisit un polynôme aléatoire $r \in R_p$

— calcule $c = m + p.r \times h \mod q$ qui est le chiffré.

Déchiffrement

Alice :

— prend sa clé clé privée $(f_p = f^{-1} \mod p, f)$;

— calcule $fc = fm + p.rg \mod q$, (on suppose que $fm + p.rg \mod q = fm + p.rg$)

— calcule $(fc \mod q) \mod p = fm \mod p$

— calcule $f_p(fc \mod q) \mod p = f_p fm \mod p = m \mod p = m$

3.2 Signature

3.2.1 Modélisation des signatures numériques

Une signature est un procédé, qui, appliqué à un message, garantit la non répudiation par le signataire et donc réalise les deux objectifs suivants :

— identification unique du signataire,

— et preuve d'accord sur le contenu du document.

Elle doit posséder les propriétés suivantes :

1. unique
2. impossible à usurper
3. impossible à répudier par son auteur,
4. facile à vérifier par un tiers,

5. facile à générer

Définition

Un système de signature est composé d'un quintuplet $(\mathcal{P}, \mathcal{S}, S_{s_K}, V_{p_K}, \mathcal{K})$ où :

1. $\mathcal{P}$ est un ensemble appelé espace des textes à signer ;
2. $\mathcal{S}$ est un ensemble appelé espace des signatures ;
3. $\mathcal{H}$ une fonction de hachage (éventuel) ;
4. $\mathcal{K}$ ensemble des paires de clés (p_K, s_K) où p_K clé publique, s_K clé privée ;
5. $S_{s_K} : \mathcal{P} \to \mathcal{S} : m \mapsto S_{s_K}(H(m)) = \sigma_K$ la signature de m avec la clé privée s_K.
6. $V_{p_K} : \mathcal{P} \times \mathcal{P} \to \{0,1\} = \{V, F\} : (m, \sigma) \mapsto V_{p_K}(m)$ qui vaut 1 si la signature est valide et 0 sinon.

NB : Ici, on a une signature avec appendice, c'est-à-dire on utilise m dans la vérification ; l'autre possibilité ce sont les signatures "with message recovery", c'est-à-dire qu'on utilise pas m lors de la vérification mais on le déduit de la procédure de vérification.

3.2.2 Signature El Gamal 1976

On considère G un groupe, H un sous groupe de G de cardinal $\#H = q$, g un générateur de H. On considère une fonction de hachage h éventuellement.

Alice : clé privée/publique $(a, y = g^a)$;

Signature d'Alice :

— choisir $k < q \ / \ k \wedge q = 1$, calculer $r = g^k$;

— calculer $k^{-1} \mod q,\ h(m),\ et\ h(r)$;

— calculer $s = k^{-1}[h(m) - ah(r)] \mod q$;

— signature : (r, s).

Vérification par Bob :

— prendre la clé publique g^a d'Alice ;

— calculer $h(m),\ et\ h(r)$;

— calculer $V_1 = y^{h(r)} r^s$;

— calculer $V_2 = g^{h(m)}$;

— accepter la signature si et seulement si $V_1 = V_2$.

Chapitre 4

Notions de base sur la sécurité prouvée

Les preuves de sécurité en cryptographie dépendent du système de cryptographie :

— cryptographie symétrique idéale : c'est la sécurité inconditionnelle (la puissance de calcul de l'attaquant est illimitée) ;

— cryptographie non symétrique : c'est la sécurité réductionniste (on transforme toute réussite d'une attaque en la solution d'une instance d'un problème réputé difficile).

4.1 Sécurité inconditionnelle

La théorie Shannon a permis introduire la notion de sécurité parfaite (sécurité inconditionnelle) . Dans cette théorie, le système est utilisé de la manière suivante : "à chaque nouveau chiffrement d'un texte clair, une nouvelle clé est utilisée (elle doit être aléatoire et aussi longue que le message à chiffrer)". Un exemple d'un tel chiffrement est le chiffrement de Vernam ou "One Time Pad". Dans la théorie de l'information de Shannon appliquée à la cryptographie, un cryptosystème est cryptographiquement sûr si la probabilité d'avoir une information claire ne varie pas même si on connaît son chiffré c'est à dire en terme de probabilité $P(x/y) = P(x)\ \forall x, y$ tel que $y = E(x)$ où E est la fonction de chiffrement (sauf la longueur évidemment car en général x et y ont même longueur : $|x| = |y|$). Les résultats de la théorie de l'information de Shannon pour la cryptographie peuvent être résumés dans les deux théorèmes importants suivants :

Théorème 4.1.1. *Si on suppose que* $\forall y \in \mathcal{C},\ P(y) > 0$ *(où* P *est une probabilité sur* $\mathcal{C}$ *et*

$\mathcal{P}$) et tel que le système est parfaitement sûr, alors $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{P}|$ où $\mathcal{K}$, $\mathcal{C}$, $\mathcal{P}$ représentent l'espace des clés, des chiffres et des clairs respectivement.

Théorème 4.1.2. *Soit un système cryptographique vérifiant $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$ ainsi $P(y) > 0$, $\forall y \in \mathcal{C}$. Ce système est à sécurité parfaite si et seulement si les conditions suivantes sont réalisées :*

a) toutes les clés sont équiprobables ;

b) pour chaque $x \in \mathcal{P}$ et chaque $y \in \mathcal{C}$, il existe une unique clé $k \in \mathcal{K}$ tel que $e_k(x) = y$ où e_k est l'algorithme de chiffrement.

Ainsi pour qu'un système symétrique soit à sécurité parfaite, il faut au moins que :

c_1) les clés soient aussi longue que le message à chiffrer : chaque bit d'information du message doit être chiffré par un bit de la clé ;

c_2) les clés sont équiprobables c'est à dire fabriquées aléatoirement ;

c_3) chaque clé doit être utilisée une et une seule fois (et chaque message doit être chiffré une seule fois). Mais il faut au préalable que les messages et les clés aient une taille suffisamment grande (≥ 160 *bits*)

Noter que tout algorithme satisfaisant aux critères Shannon est équivalent au "One Time Pad" de Vernam.

Ainsi, si ces conditions sont respectées quel que soit la puissance de l'attaquant, il ne peut rien faire d'où la vocable de "sécurité inconditionnelle ou parfaite" qui signifie que "l'attaquant ne peut tirer aucune information sur le texte chiffré.

Mais cette sécurité parfaite a deux inconvénients :

— elle est trop forte pour les applications civiles à cause de la longueur des clés et leur caractère aléatoire ;

— les algorithmes avec cette sécurité sont équivalents au "One Time Pad" et donc relèvent de la cryptographie à clé secrète qui ne couvre pas donc certains besoins de sécurité pratique comme la non répudiation.

On affaiblit cette notion en imposant réellement l'impossibilité pour l'adversaire de calculer en pratique quelques informations à partir du chiffré qu'on ne peut obtenir du clair seul.

Ainsi, on peut supposer que l'attaquant a accès plusieurs outils tels que le fait de pouvoir faire de chiffrer le chiffré de son choix (en dehors du chiffré qu'il veut attaquer appelé le challenge). A partir de là il va falloir définir des modèles d'adversaires en fonction de ce que l'attaquant peut raisonnablement calculer et de ce dont il dispose pour tenter ce calcul. Ce modèle de sécurité est celui de la cryptographie asymétrique.

4.2 Sécurité réductionniste

Comme cela a été annoncé précédemment, les notions de sécurité utilisées ici prennent en compte deux aspects :

— ce qu'on exige être impossible à calculer en temps raisonnable ;

— ce qu'on suppose à la disposition de l'attaquant pour tenter ce calcul.

Modèle réductionniste :

"Réduire le succès d'une attaque en la solution d'une instance considérée comme difficile comme un problème donné".

En pratique :

"Si l'on suppose l'existence d'un algorithme A qui casse en pratique les instances de taille t d'un système S, on en déduit qu'on peut construire un algorithme B qui casse une instance de taille t' d'un problème D, réputé difficile".

Cette technique de réduction telle que présentée ici n'est intéressante que lorsque le temps de construire B sous l'hypothèse de l'existence de A est polynômial (=temps de réduction). Dans la pratique de la modélisation il y a deux approches :

— **modèle standard** : on réduit comme ci-dessus la sécurité du protocole ou de l'algorithme à celui de la difficulté d'un problème tel que la factorisation, RSA, le logarithme discret, etc ... .

— **modèle de l'oracle aléatoire** : on suppose l'existence de fonction parfaitement aléatoire (qu'on implémente via des fonctions de hachage dans les systèmes de cryptographie).

4.3 Modèle de l'oracle aléatoire

- **Oracle :** c'est un algorithme (machine Turing ou une fonction f) auquel on peut soumettre une entrée x et recevoir une sortie $f(x)$ et que cet algorithme se comporte comme une boîte noire pour le requérant c'est à dire que le requérant n'exécute pas lui même f mais a simplement accès à tout $f(x)$ correspondant au x de son choix.

Généralement on peut supposer que :

- le requérant ne connaît pas une expression algébrique (ainsi que tous les paramètres utilisés) de la fonction comme dans le cas d'une fonction de déchiffrement ; ainsi quand un requérant est capable de faire déchiffrer le chiffré de son choix sans forcément savoir comment on déchiffre, on dit qu'il a accès à un oracle de déchiffrement ;

- le requérant connaît une expression algébrique de la fonction mais les sorties de cette fonction se comportent de façon aléatoire, on dit que c'est un oracle aléatoire (c'est le cas d'une fonction de hachage idéalisée).

Fonctions aléatoires

On considère $\mathcal{P} = \{f : \{0,1\}^* \longrightarrow \{0,1\}^\infty\}$, f prend en entrée un mot de longueur finie et produit une suite de longueur infinie.

Une fonction $f \in \mathcal{P}$ est dite prise au hasard c'est à dire aléatoire si pour tout $x \in \{0,1\}^*$, chaque bit de la suite infinie $f(x)$ est prise au hasard c'est à dire $p(0) = p(1) = 1/2$.

Oracle aléatoire :

"Un oracle aléatoire est une procédure qui permet pour $x \in \{0,1\}^*$ de produire $f(x)$ où $f \in \mathcal{P}$ est une fonction aléatoire sans révéler d'aucune façon le procédé de calcul."

1) Les oracles sont utilisées dans les preuves de sécurité : par exemple si un attaquant ne peut casser un chiffré même s'il a accès à un oracle de déchiffrement alors cela signifie qu'il n'a pas mieux à faire pour trouver le clair correspondant au chiffré qu'il veut attaquer (appelé challenge) que de choisir le clair au hasard.
2) Si f est un oracle aléatoire, on peut définir une fonction aléatoire ou une fonction de hachage $g : \{0,1\}^k \longrightarrow \{0,1\}^s$, en posant $g(x) =$ les s premiers bits de $f(x)$ pour $x \in \{0,1\}^k$;
3) Si $f : \{0,1\}^* \longrightarrow \{0,1\}^\infty$ est un oracle aléatoire et $\varphi : \{0,1\}^\infty \longrightarrow \{0,1\}^\infty$ est une

bijection c'est à dire une renumérotation alors $\varphi \circ f$ est un nouveau oracle aléatoire.

Modèle Oracle aléatoire vs Modèle standard

Dans une preuve de sécurité, si on fait appels à des oracles aléatoires, on dit que la preuve se fait dans le modèle de l'oracle aléatoire (dans ce cas ces oracles aléatoires correspondent à des fonctions de hachage dans le cryptosystème). Si on utilise pas d'oracle aléatoire, on dit qu'on est dans le modèle standard.

Signification d'une preuve dans le modèle oracle aléatoire

Les preuves dans le modèle oracle aléatoire sont développées dans M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Communications Security, ACM, 1993.

Il est connu qu'une preuve dans le modèle oracle aléatoire n'implique pas que le système est sûr dans la vie pratique : voir l'étude de R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology*, revisited, STOC 98, ACM, 1998.

Néanmoins, il est largement accepté par la communauté scientifique, la fait de construire des algorithmes avec des preuves de sécurité dans le modèle oracle aléatoire.

Chapitre 5

Objectifs de sécurité et Modèles d'adversaires en chiffrement à clés publiques

5.1 Les objectifs généraux de sécurité du chiffrement à clé publique

chiffrement à clé publique

Algorithme de génération de clés : $\mathcal{K}$

c'est un algorithme probabiliste qui prend en entrée un paramètre de sécurité k et sort une paire de clé (p_k, s_k) cohérente où p_k est la clé publique et s_k la clé privée. Cet algorithme s'exécute en temps maximum polynômial, notation : $(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$

Algorithme de déchiffrement : $\mathcal{D}$

$\mathcal{D}$ est un algorithme déterministe qui prend en entrée une clé privée s_k et un chiffré $y \in \{0,1\}^*$ et qui sort le texte clair si y est un chiffré valide ou un symbole d'erreur $\perp$ si y n'est pas un chiffré valide.

Cet algorithme s'exécute en temps polynômial, notation :

$x' \longleftarrow \mathcal{D}(s_k, y)$ avec $x' \in \{x, \perp\}$

Algorithme de chiffrement : $\mathcal{C}$

c'est un algorithme probabiliste $\mathcal{E}$ qui prend en entrée la clé publique p_k et un texte clair $x \in \mathcal{P} \subseteq \{0,1\}^*$ (où $\mathcal{P}$ est l'ensemble des textes clairs) et qui sort le chiffré $y \in \mathcal{C} \subseteq \{0,1\}^*$ (où $\mathcal{C}$ est l'ensemble des chiffrés).

Cet algorithme s'exécute en temps polynômial $r \longleftarrow R$

Notation : $y \longleftarrow \mathcal{E}(p_k, x, r)$ où r est une valeur aléatoire et R un générateur aléatoire.

Objectifs généraux de sécurité

Pour le chiffrement à clé publique les objectifs généraux de sécurité sont les suivants :

1. **"Incassabilité" :** Il doit être possible de déduire la clé secrété de la clé publique.
2. **Fonction à sens unique :** La fonction de chiffrement doit etre à sens unique, donc on doit pas pouvoir déduire un clair de son chiffré (Diffie-Hellman 1976).

   On considère l'expérience qui consiste à prendre en entré les données publiques dont un chiffré c d'un message m et à produire un message $\hat{m}$ puis à sortir 1 si $m = \hat{m}$ et 0 si non. Le sucés de l'expérience précédente est $\text{Succ}^{OW}(\mathcal{A}, k) = Pr(\text{Exp}^{OW}(\mathcal{A}, k) = 1)$ où

   $$Pr(\text{Exp}^{OW}(\mathcal{A}, k) = 1) = Pr\left(m = \widehat{m} / \begin{array}{l}(p_k, s_k) \longleftarrow \mathcal{K}(1^k),\ c \longleftarrow \mathcal{C}(p_k, m, r) \\ r, r' \longleftarrow R,\ m \longleftarrow \mathcal{P},\ \widehat{m} \longleftarrow \mathcal{A}(p_k, r', c)\end{array}\right)$$

   Généralement on omet r' car on sait que le chiffrement est probabiliste et on écrit

   $$Pr(\text{Exp}^{OW}(\mathcal{A}, k) = 1) = Pr\left(m = \widehat{m} / \begin{array}{l}(p_k, s_k) \longleftarrow \mathcal{K}(1^k),\ c \longleftarrow \mathcal{C}(p_k, m, r) \\ r \longleftarrow R,\ m \longleftarrow \mathcal{P},\ \widehat{m} \longleftarrow \mathcal{A}(p_k, r, c)\end{array}\right)$$

   Pour que la fonction soit à sens unique il faut que $\text{Exp}^{OW}(\mathcal{A}, k) = 1)$ soit négligeable en fonction du paramètre de sécurité k c'est-à-dire $\text{Exp}^{OW}(\mathcal{A}, k) = 1) \leq \frac{1}{f(k)}$ où f est un polynôme
3. **Indistinguabilité :** Connaissant le chiffré c d'un clair m, on doit pas pouvoir déduire un seul bit d'information de m (d'une façon meilleure qu'un choix au hasard) (Introduit par Goldwasser et Micali 1984)
4. **Non-malléabilité :** (absence de corrélation) Il doit être impossible de transformer un chiffré c_1 d'un clair m_1 en un chiffré c_2 d'un clair m_2 de sorte que m_1 et m_2 soient reliés avec une probabilité meilleur que la probabilité uniforme (Dolev, Dwork, Naor 1991)

On notera que par définition :

Non-malléabilité $\rightarrow$ Indistinguabilité $\rightarrow$ Fonction à sens unique $\rightarrow$ Incassabilité

5.2 Avantage de l'attaquant

L'attaquant est modélisé par un algorithme $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ composé de deux algorithmes : $\mathcal{A}_1$ s'exécute avant la réception du challenge C^* et $\mathcal{A}_2$ après. Noter challenge C^* est le chiffré qu'on souhaite soumettre à l'attaquant pour qu'il l'utilise pour mettre à défaut un des objectifs de sécurité cités précédemment)

Chaque $\mathcal{A}_i$ peut faire appelle à un oracle $\mathcal{O}_i$ et un algorithme probabiliste polynômial. La sortie de l'algorithme $\mathcal{A}$ est celle de $\mathcal{A}_2$ et son entrée est celle de $\mathcal{A}_1$.

L'attaquant sera efficace si sa probabilité de succès est significativement plus forte que la probabilité de succès d'une réponse donnée avec le maximum de vraisemblance (sans connaître le challenge C^* et sans avoir accès à un oracle de déchiffrement).

Avantage de l'attaquant :

Soit MV un algorithme qui répond avec le maximum de vraisemblance $|\Pr(\mathcal{A}_2 = \text{bonne réponse}) - Pr(MV = \text{bonne réponse})|$ est l'avantage de l'attaquant.

Dans le cas où les réponses de MV sont parfaitement aléatoires, on a : $\Pr(\text{MV} = \text{bonne réponse}) = 1/2$ et l'avantage est :

$$\text{Adv}(\mathcal{A}) = 2|Pr(\mathcal{A}_2 = \text{bonne réponse}) - 1|$$

5.3 Modèles d'adversaires

Attaque à textes clairs choisis (chosen plaint text attack : CPA)

L'attaquant peut obtenir les chiffrés de son choix. Par exemple c'est un cas trivial en cryptographie à clé publique car la clé publique est accessible.

Attaque non adaptative à chiffré choisis (nonadaptative chosen cipher text attack : CCA 1)

En plus de la clé publique, l'attaquant a accès à un oracle de déchiffrement ($\mathcal{O}_1$) uniquement avant de recevoir le challenge c'est à dire le chiffré à attaquer (donc il ne pourra pas

adapter ses requêtes à l'oracle en fonction des informations qu'il a sur le challenge).

Attaque adaptative à chiffré choisis (adaptative chosen cipher text attack : CCA 2)

En plus de la clé publique, l'attaquant peut accéder à un oracle de déchiffrement avant ($\mathcal{O}_1$) et après ($\mathcal{O}_2$) avoir reçu le challenge. Mais le challenge n'est pas soumis à l'oracle. Cette attaque est plus forte que les précédentes car après avoir pris connaissance du challenge il peut adapter les chiffrés qu'il souhaite faire déchiffrer.

Nota bene C'est l'utilisation ou non d'oracle de déchiffrement qui fait la différence.

On note $\mathcal{O}_i = \mathcal{O}$ l'usage de l'oracle $\mathcal{O}$ pour $i = 1, 2$ et quand on utilise pas d'oracle on met $\mathcal{O}_i = \emptyset$.

Formalise attaquant	$\mathcal{A}_1$	$\mathcal{A}_2$
CPA	$\mathcal{O}_1 = \emptyset$	$\mathcal{O}_1 = \emptyset$
CCA 1	$\mathcal{O}_1 = \mathcal{O}$	$\mathcal{O}_2 = \emptyset$
CCA 2	$\mathcal{O}_1 = \mathcal{O}$	$\mathcal{O}_2 = \mathcal{O}$

Chapitre 6

Formalisme des preuves de sécurité pour le chiffrement à clés publique

Introduction

Il existe plusieurs types de preuves de sécurité :

— Indistinguabilité ;

— Sécurité sémantique ;

— Non malléabilité basée sur la comparaison ;

— Non malléabilité basée sur la signature.

Dans ce qui suit on va définir les différents concepts ci-dessus puis étudier les relations entre eux.

6.1 Sécurité sémantique

La sécurité sémantique stipule qu'un attaquant ne peut tirer aucune information du chiffrer même s'il connaît un ensemble fini de textes clairs M (ces textes clairs sont les déchiffrés de chiffrés connus).

a)

Modélisation :

Expérience :

$\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ est un système à clé publique dont le paramètre de sécurité est noté k.

On note $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ l'algorithme de l'attaquant.

Etape 1 :

Le challenger déroule l'algorithme $\mathcal{K}$ pour générer une paire de clés (p_k, s_k) en prenant pour entrée 1^k : $(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$, une copie de la clé publique est envoyée à l'attaquant.

Etape 2 :

$\mathcal{A}_1$ prend en entrée la clé p_k, utilise éventuellement un oracle de déchiffrement $\mathcal{O}_1$ et sort la description d'un ensemble $\mathcal{M}$ de textes clairs (muni de la probabilité uniforme) ainsi que la description de l'état S du système (S peut représenter tous les infos et outils dont l'attaquant dispose et souhaite utiliser dans $\mathcal{A}_2$.

Etape 3 :

Un oracle de chiffrement aléatoire $\mathcal{O}_c$ choisit un texte clair $m \in \mathcal{M}$ et le chiffre.

$\mathcal{O}_c$:

Entrée : $\mathcal{M}, p_k, R$

$m \longleftarrow \mathcal{M}$

$r \longleftarrow R$

$c \longleftarrow \mathcal{E}(p_k, m, r)$

sortie : c (=le challenge).

Etape 4 :

$\mathcal{A}_2$ prend en entrée $(\mathcal{M}, s, c)$ et éventuellement un oracle de déchiffrement $\mathcal{O}_2$ et sort une valeur z et la description d'une fonction $g : \mathcal{M} \longrightarrow \mathcal{M}$ qui évalue la relation entre m et z.

NB : l'attaquant souhaite que $g(m) = z$ avec une grande probabilité.

Modélisation des 4 étapes :

$\mathrm{Esp}_{\Pi}^{SS}(\mathcal{A}, k)$, SS= Sécurité Sémantique : Expérience normale relativement à SS (réponses basées sur tous les moyens de l'attaquant) .

$\widetilde{\mathrm{Esp}}_{\Pi}^{SS}(\mathcal{A}, k)$, version avec le maximum de vraisemblance (réponses au hasard).

Version pratique Version avec le maximum de vraisemblance

$(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$ challenger $(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$

$(\mathcal{M}, s) \longleftarrow \mathcal{A}^{\mathcal{O}_1}(p_k)$ attaquant $(\mathcal{M}', s) \longleftarrow \mathcal{A}^{\emptyset}(p_k)$

$m \overset{rand}{\longleftarrow} \mathcal{M}$ challenger $m, \widetilde{m} \overset{rand}{\longleftarrow} \mathcal{M}'$

$r \overset{rand}{\longleftarrow} R$ challenger $r \overset{rand}{\longleftarrow} R$

$c^\star \longleftarrow \mathcal{E}(p_k, m, r)$ challenger $c \longleftarrow \mathcal{E}(p_k, m, r)$

$(g, z) \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathcal{M}, s, c^\star)$ attaquant $(g, z) \overset{rand}{\longleftarrow} \mathcal{A}_2^{\emptyset}(\mathcal{M}', s, c)$

$$\left.\begin{array}{c} \widetilde{z} \longleftarrow g(m) \\ si\ z = \widetilde{z} \text{ retourner1} \\ \text{sinon retourner 0} \end{array}\right\} \longrightarrow \text{Résultats} \longleftarrow \left\{\begin{array}{c} \widetilde{z} \longleftarrow g(\widetilde{m}) \\ \text{si } z = \widetilde{z} \text{ retourner1} \\ \text{sinon retourner 0} \end{array}\right.$$

Avantage :

On compare les probabilités de succès dans deux scénarios qui doivent être indistinguable.

L'avantage de l'attaquant est

$$\mathrm{Adv}_{\Pi}^{\mathrm{SS}}(A, k) = \left|\Pr\left[\mathrm{Exp}_{\Pi}^{\mathrm{SS}}(A, k) = 1\right] - \Pr\left[\mathrm{Exp}_{\Pi}^{\mathrm{SS}}(A, k) = 1\right]\right|$$

6.1.1 Définition de la sécurité sémantique

Définition :

Le système de cryptographie à clé publique $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ est sémantiquement sûr contre une attaque du type CPA (respectivement CCA1, CCA2) si et seulement si pour tout polynôme $P(k)$ (où k est le paramètre de sécurité) et pour tout attaquant $\mathcal{A}$ de classe CPA (respectivement : CCA1, CCA2) dont les entrées et sorties se comportent comme dans $\mathrm{Exp}_{\Pi}^{\mathrm{SS}}(A, k)$ et $\widetilde{\mathrm{Exp}}_{\Pi}^{\mathrm{SS}}(A, k)$ et tel que :

- $\mathcal{A}$ s'exécute en temps $\leq P(k)$;
- l'échantillonnage de $\mathcal{M}$ s'exécute en temps $\leq P(k)$;
- la fonction g est calculable en temps $P(k)$ et pour tout autre $m > 0$, $\lim k^m \mathrm{Adv}_{\Pi}^{\mathrm{SS}}(A, k) = 0$;

Autre approche équivalente :

"Pour extraire une information $g(m)$ d'un message m connaissant une autre information $h(m)$, on fait pas mieux si on connaît le chiffré c de m". Ce qu'on formalise par :

$$\mathrm{Adv}_{\Pi}(\mathcal{A}) = \left|\Pr\left[\mathcal{A}\big(h(x_k), \mathcal{E}(\mathcal{K}(1^k))(x_k)\big) = g(x_k)\right] - \Pr\left[\mathcal{A}(h(x_k)) = g(x_k)\right]\right|$$

est négligeable en k (=paramètre de sécurité) où x_k est une variable aléatoire sur l'ensemble des messages et $\mathcal{K}(1^k))(x_k)$ est le chiffré de x_k.

6.2 Indistinguabilité (dichotomie)

6.2.1 Modélisation :

Etant donnés

- deux messages m_0 et m_1 ;

- c le chiffré de l'un des m_i (au hasard) ;

l'attaquant ne doit pas pouvoir distinguer lequel des m_i a été chiffré en c.

Ainsi la probabilité de réussite de l'attaquant doit être proche de celui aurait répondu au hasard.

Comme la réponse au hasard pour un message sur deux a pour probabilité 1/2 alors l'avantage de l'attaquant est Adv = Pr(A = bonne réponse) − 1/2 ou Adv = 2Pr(A = bonne réponse) − 1 car on s'intéresse à l'écart et non à la valeur de Pr(A = bonne réponse).

$\mathrm{Esp}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k)$, IND= Indistinguabilité : Expérience normale relativement à IND .

$\widetilde{\mathrm{Esp}}_{\Pi}^{SS}(\mathcal{A}, k)$, version avec le maximum de vraisemblance qui correspondra ici avec une expérience parfaitement aléatoire.

Modélisation du scénario

$(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$ Challenger

$(m_0, m_1, s) \longleftarrow \mathcal{A}^{\theta_1}(p_k)$ Attaquant

$b \overset{rand}{\longleftarrow} \{0, 1\}$ Challenger

$r \overset{rand}{\longleftarrow} R$ Challenger

$c \longleftarrow \mathcal{E}(p_k, m_b, r)$ Challenger

$\widetilde{b} \longleftarrow \mathcal{A}_2(m_0, m_1, c, s)$ Attaquant

$\left.\begin{array}{c} \widetilde{b} \longleftarrow \{0, 1\} \\ \text{si } b = \widetilde{b} \text{ retourner1} \\ \text{sinon retourner 0} \end{array}\right\} \longrightarrow$ Résultats

Avantage :

$$\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k) = 2\Pr\left[Exp_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k) = 1\right] - 1$$

On note souvent pour résumer

$$\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k) = 2\Pr\left[b = \widetilde{b}\right] - 1$$

6.2.2 Définition de l'indistinguanilité

Définition

On dit que le système cryptographique Π est sûr contre l'indistinguabilité contre une attaque de type CPA (respectivement : CCA1, CCA2) si et seulement pour tout attaquant $\mathcal{A}$ de la classe CPA (respectivement : CCA1, CCA2) dont les entrées et sorties se comportent comme il a été décrit dans $\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k)$ et pour tout entier $m > 0$, nous avons $\lim_{k \longrightarrow +\infty} \mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k) = 0$.

Autre approche équivalente :

On considère trois messages (m_0, m_1, c)

L'attaquant

- connaissant $\mathcal{E}(p_k, m_0, r_0) = c_0$, déterminer la probabilité que c soit le chiffré de m_0 c'est à dire $\mathcal{E}(p_k, m_0, r_0') = c$,

$$\Pr\left[c = \mathcal{E}(p_k, m_0, r_0')/\mathcal{E}(p_k, m_0, r_0)\right]$$

- connaisssant $\mathcal{E}(p_k, m_1, r_1) = c_1$, déterminer

$$\Pr\left[c = \mathcal{E}(p_k, m_1, r_1')/\mathcal{E}(p_k, m_1, r_1)\right]$$

Si ces deux probabilités sont suffisamment distinctes alors l'attaquant peut réussir à distinguer lequel des m_i $(i = 0\ ou\ 1)$ est chiffré en c.

On écrit alors les deux expériences suivantes.

Etant donné un triplet (m_0, m_1, c), l'attaquant $\mathcal{A}$ déroule les deux expériences :

$Exp_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k, (m_0, c))$ $\qquad$ $\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, t, (m_1, c))$

$r_0 \longleftarrow R$ $\quad$ $r_1 \longleftarrow R$

$c_0 \longleftarrow \mathcal{E}(p_k, m_0, r_0)$ $\quad$ $c_1 \longleftarrow \mathcal{E}(p_k, m_1, r_1)$

$b \longleftarrow \mathcal{A}(c, c_0)$ $\quad$ $\widetilde{b} \longleftarrow \mathcal{A}(c, c_1)$

$b \in \{0,1\}$ $\quad$ $\widetilde{b} \in \{0,1\}$

Donc pour qu'il y ait indistinguabilité, il faut que $\forall m > 0$,

$$\lim k^m Pr\left[(m_0, m_1, c)/\left(\Pr(\mathcal{A}(c, \mathcal{E}(p_k, m_0, r_0))) = 1\right) - \left(\Pr(\mathcal{A}(c, \mathcal{E}(p_k, m_1, r_1))) = 1\right)\right] = 0$$

ou en terme de variables aléatoires

$$\lim k^m \Pr\left[(X_n, Y_n, Z_n)/\left(\Pr(\mathcal{A}(Z_n, \mathcal{E}(\mathcal{K}_{1^k}, X_n))) = 1\right) - \left(\Pr(\mathcal{A}(Z_n, \mathcal{E}(\mathcal{K}_{1^k}, Y_n))) = 1\right)\right] = 0$$

6.3 La non malléabilité :

6.3.1 Modélisation de la Non malléabilité par comparaison (CNM) :

Intuitivement cette notion stipule l'impossibilité de construire un chiffré c' d'un clair m' sachant un chiffré $c^\star$ d'un clair m de sorte qu'une relation R connue relie m et m'. On note la relation $m'R(m)$ où $m' = R(m)$ ou $R(m, m')$.

Plus généralement, l'attaquant souhaite à partir du chalenge $c^\star$, produire une relation R et un ensemble Y de textes chiffrés ne contenant pas $c^\star$ de sorte que si on déchiffre Y en X, alors m soit relié à X à travers R (ce qu'on note $R(m, X)$) plus fréquemment, autrement dit que la probabilité de $R(m, X)$, est plus significative que celle de $R(\tilde{m}, X)$ où $\tilde{m}$ est tiré au hasard.

NB :

$\mathrm{Exp}_{\Pi}^{\mathrm{CNM}}(\mathcal{A}, k)$ $\quad$ $\widetilde{\mathrm{Exp}}_{\Pi}^{\mathrm{CNM}}(\mathcal{A}, k)$

$(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$ $\quad$ $(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$

$(\mathcal{M}, s) \longleftarrow \mathcal{A}_1^{\mathcal{O}_1}(p_k)$ $\quad$ $(\mathcal{M}, s) \longleftarrow \mathcal{A}^{\emptyset}(p_k)$

$m \overset{rand}{\longleftarrow} \mathcal{M}$ $\quad$ $m, \widetilde{m} \overset{rand}{\longleftarrow} \mathcal{M}$

$r \overset{rand}{\longleftarrow} R$ $\quad$ $r \overset{rand}{\longleftarrow} R$

$c^\star \longleftarrow \mathcal{E}(p_k, m, r)$ $\quad$ $c \longleftarrow \mathcal{E}(p_k, m, r)$

$(R, Y) \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathcal{M}, s, c^\star)$
$c^\star \notin Y$
$X \longleftarrow \mathcal{D}(Y, s_k)$
$m \notin X$
Si $R(m, X)$ retourner 1
Sinon retourner 0

$(R, Y') \xleftarrow{rand} \mathcal{A}_2^{\emptyset}(\mathcal{M}, s, c)$
$c \notin Y'$
$X' \longleftarrow \mathcal{D}(Y', s_k)$
$m \notin X'$
Si $R(\widetilde{m}, X')$ retourner 1
Sinon retourner 0

6.3.2 Modélisation de la Non malléabilité par simulation (SNM) :

- On fixe une relation à 4 arguments $R(m, X, \mathcal{M}, z)$ où m est un texte clair, X un ensemble de textes clairs (X de petite taille), $\mathcal{M}$ ensemble des textes clairs pour lequel, on dispose d'un algorithme d'échantillonnage et un mot d'état z.

- $S = (S_1, S_2)$ est l'algorithme simulateur qui fait ce que devrait faire l'attaquant en répondant au hasard et en utilisant le moins d'informations possibles, donc S ne connaît pas le challenge et n'utilise pas l'oracle de déchiffrement.

$\mathrm{Exp}_{\Pi}^{\mathrm{SNM}}(\mathcal{A}, k)$
$(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$
$(\mathcal{M}, z_1, z_2) \longleftarrow \mathcal{A}^{\mathcal{O}_1}(p_k)$
$m \xleftarrow{rand} \mathcal{M}$
$r \xleftarrow{rand} R$
$c^\star \longleftarrow \mathcal{E}(p_k, m, r)$
$Y \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathcal{M}, z_2, c^\star)$
$c^\star \notin Y$
$X \longleftarrow \mathcal{D}(Y, s_k)$
Si $R(m, X, \mathcal{M}, z_1)$ retourner 1
Si non retourner 0

$\widetilde{\mathrm{Exp}}_{\Pi}^{\mathrm{SNM}}(\mathcal{A}, k)$
$(p_k, s_k) \longleftarrow \mathcal{K}(1^k)$
$(\mathcal{M}, z_1, z_2) \longleftarrow S_1^{\emptyset}(p_k)$
$m' \xleftarrow{rand} \mathcal{M}$
$c \longleftarrow \mathcal{E}(p_k, m', r)$
$Y' \longleftarrow S_2^{\emptyset}(\mathcal{M})$
$X' \longleftarrow \mathcal{D}(Y', s_k)$
Si $R(m', X', \mathcal{M}, z_2)$ retourner 1
Si non retourner 0

6.3.3 Définition des deux types de malléabilité

Définition 6.3.1. .

1. ***Non malléabilité basée sur la comparaison (CNM) :***

Le système cryptographique Π *est sûr pour la non malléabilité basée sur la comparaison contre tout attaque de type CPA (resp. CCA1, CCA2) si et seulement si pour tout polynôme* $P(k)$*, pour tout attaquant de la classe CPA (resp. CCA1, CCA2) dont les entrées et sorties se comportent comme il a été indiqué dans* $Exp_{\Pi}^{\mathrm{CNM}}(\mathcal{A}, k)$ *et* $Exp_{\Pi}^{\mathrm{CNM}}(\mathcal{A}, k)$ *et tels que :*

(a) $\mathcal{A}$ *s'exécute en temps* $\leq P(k)$ *;*

(b) l'échantillonnage de $\mathcal{M}$ *s'exécute en temps* $\leq P(k)$ *;*

et pour tout entier $m > 0$*, nous avons* $\lim_{k\to\infty} k^m Adv_{\Pi}^{\mathrm{CNM}}(\mathcal{A}, k) = 0$.

2. ***Non malléabilité basée sur la simulation (SNM) :***

Le système cryptographique Π *est sûr pour la non malléabilité basée sur la simulation contre tout attaque de type CPA (resp. CCA1, CCA2) si et seulement si pour toute relation calculable en temps* $\leq P(k)$*, pour tout attaquant de la classe CPA (resp. CCA1, CCA2) dont les entrées et sorties se comportent comme il a été indiqué dans* $Exp_{\Pi}^{\mathrm{SNM}}(\mathcal{A}, R, k)$ *et* $Exp_{\Pi}^{\mathrm{SNM}}(\mathcal{A}, S, k)$ *et tels que :*

(a) $\mathcal{A}$ *s'exécute en temps* $\leq P(k)$ *;*

(b) l'échantillonnage de $\mathcal{M}$ *s'exécute en temps* $\leq P(k)$ *;*

(c) il existe un algorithme S *qui s'exécute en temps polynômial et qui produit un* $\mathcal{M}'$ *dont échantillonnage est en temps polynômial et tels que pour tout* $m > 0$*,* $\lim_{k\to\infty} k^m Adv_{\Pi}^{\mathrm{SNM}}(\mathcal{A}, S, R, k) = 0$.

NB1 : Noter que les algorithmes non probabilisés ne sont pas sémentiquement sûr raison pour laquelle on ne travaille qu'avec des algorithmes probabilistes dans les preuves de sécurité. Dans la suite pour simplifier on omettra $r \longleftarrow R$ dans la liste des expressions.

NB2 :

- La sécurité sémantique et l'indistinguabilité sont équivalentes ;

- La non malléabilité basée sur la comparaison et celle basée sur la simulation sont équivalentes.

Voir l'article le cours D. Pointcheval *Provable Security for Public Key Schemes* Advanced Course on Contemporary Cryptology, pages 133-189, June 2005. http ://www.di.ens.fr/ pointche/pub.php ?reference=Po04

6.4 Diagramme des relations entre les différentes notions de sécurité

$$\begin{array}{ccccc} NM-CPA & \longleftarrow & NM-CCA1 & \longleftarrow & NM-CCA2 \\ \downarrow & & & & \updownarrow \\ IND-CPA & \longleftarrow & IND-CCA1 & \longleftarrow & IND-CCA2 \end{array}$$

Proposition 6.4.1. *Soit $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ un attaquant sur un système de cryptographie Π dans le modèle IND. Alors il existe un autre attaquant $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ sur Π dans le modèle IND tel que*

- *les messages de même longueur, que $\mathcal{B}_1$ produit, soit toujours distincts ;*
- *$Adv_{\Pi}^{IND}(\mathcal{A}, k) = Adv_{\Pi}^{IND}(\mathcal{B}, k)$;*
- *le temps de calcul de $\mathcal{B}$ est égal à celui de $\mathcal{A}$ à un facteur constant près.*

Preuve :

1. Construire des messages distincts

   Connaissant $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, il faut construire $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$

Algo : $\mathcal{B}_1^{\mathcal{O}_1}(p_k)$	Algo : $\mathcal{B}_2^{\mathcal{O}_2}(m'_0, m'_1, S, c)$
- $(m_0, m_1, s) \longleftarrow \mathcal{A}_1^{\mathcal{O}_1}(p_k)$	si $d = 0$ alors
- si $m_0 \neq m_1$ alors $d \longleftarrow 0$	$e \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(m'_0, m'_1, s)$
sinon $d \longleftarrow 1$	$e \in \{0,1\}$
- $m'_0 \longleftarrow m_0$ et $s' \longleftarrow s\|\|d$	sinon
si $d = 0$ alors $m'_1 \longleftarrow m_1$	$e \overset{rand}{\longleftarrow} \{0,1\}$
sinon alors $m'_1 \longleftarrow \overline{m}_0$	Retourner e
Retouner (m'_0, m'_1, s)	

- On a toujours $m'_0 \neq m'_1$;

- Si $d = 1$ alors $m_0 = m_1$ et pour que l'attaquant n'ait pas d'avantage, on choisit un bit arbitraire qu'on met dans e.

2. Temps de calcul

   Toutes les expériences effectuées par $\mathcal{B}_1$ et $\mathcal{B}_2$ sont élémentaires (comparaison, affectation, concaténation, complémentaire) et en nombre fini donc le temps de calcul de $\mathcal{B}$ est similaire à celui de $\mathcal{A}$ à une constante près.

3. Montrons que $\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k) = Adv_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k)$

$$\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k) = \left\{ \begin{array}{l} (p_k, s_k) \longleftarrow \mathcal{K}(1^k); (m_0, m_1, s) \longleftarrow \mathcal{A}_1(p_k) \\ b \longleftarrow \{0,1\}; c \longleftarrow \mathcal{E}(p_k, m_b); \widetilde{b} \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(m_0, m_1, s, c) \end{array} \right\}$$

$$\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k) = \left\{ \begin{array}{l} (p_k, s_k) \longleftarrow \mathcal{K}(1^k); (m_0, m_1, s) \longleftarrow \mathcal{A}_1(p_k) \\ (m'_0, m'_1, s') \longleftarrow \mathcal{B}_1(m_0, m_1, s); b \longleftarrow \{0,1\} \\ c \longleftarrow \mathcal{E}(p_k, m'_b); \widetilde{b} \longleftarrow \mathcal{B}_2^{\mathbf{O}_2}(m'_0, m'_1, s' = s||d, c) \end{array} \right\}$$

$\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k) = 2\mathrm{Pr}_{\mathcal{A}}(b = \widetilde{b}) - 1$

$\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k) = 2\mathrm{Pr}_{\mathcal{B}}(b = \widetilde{b}) - 1$

le reste à prouver que $\mathrm{Pr}_{\mathcal{A}}(b = \widetilde{b}) = \mathrm{Pr}_{\mathcal{B}}(b = \widetilde{b})$

Soit E l'événement "$m_0 = m_1$" $\Longleftrightarrow$ "$d = 1$".

La formule de Bayes donne

(α) $\quad \mathrm{Pr}_{\mathcal{A}}(b = \widetilde{b}) = \mathrm{Pr}_{\mathcal{A}}(b = \widetilde{b}/E) \times \mathrm{Pr}_{\mathcal{A}}(E) + \mathrm{Pr}_{\mathcal{A}}(b = \widetilde{b})\mathrm{Pr}_{\mathcal{A}}(\overline{E})$

(β) $\mathrm{Pr}_{\mathcal{B}}(b = \widetilde{b}) = \mathrm{Pr}_{\mathcal{B}}(b = \widetilde{b}/E) \times \mathrm{Pr}_{\mathcal{B}}(E) + \mathrm{Pr}_{\mathcal{B}}(b = \widetilde{b})\mathrm{Pr}_{\mathcal{B}}(\overline{E})$.

Faisons les remarques suivantes :

(a1) $\mathrm{Pr}_{\mathcal{A}}(E) = \mathrm{Pr}_{\mathcal{B}}(E)$ car E ne dépend que de $\mathcal{A}$ et reste ainsi après l'action de $\mathcal{B}$.

(a2) $\mathrm{Pr}_{\mathcal{A}}(b = \widetilde{b}/E) = 1/2$ car comme E est réalisé, $\mathcal{A}$ n'a aucune information sur le bit b à moins de le choisir aléatoirement donc c'est la même situation que de la façon dont il a choisi $\widetilde{b}$, or le choix de $\widetilde{b}$ a pour probabilité $1/2$ d'où le résultat.

(a3) $\mathrm{Pr}_{\mathcal{B}}(b = \widetilde{b}/E) = 1/2$ car comme E est réalisé $\mathcal{B}_2$ sort un bit aléatoire puisque E réalisé $\Longleftrightarrow d = 1$.

(a4) $\Pr_{\mathcal{A}}(b = \widetilde{b}/\overline{E}) = \Pr_{\mathcal{B}}(b = \widetilde{b}/\overline{E})$ car $\overline{E}$ est réalisé c'est à dire $m_0 \neq m_1$, alors $\mathcal{A}_2$ et $\mathcal{B}_2$ font exactement la même chose.

Ainsi on tire de (a1), (a2), (a3) et (a4) que le résultat cherché est acquis.

Théorème 6.4.1. *Si un système* Π *est sûr au sens de la non malléabilité pour une attaque de type CPA (resp. CCA1, CCA2) alors le même système* Π *est sûr au sens de l'indistinguabilité pour une attaque de type CPA (resp. CCA1, CCA2).*

Preuve :

Soit $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ un attaquant sur Π au sens IND.

Sachant que Π est sûr au sens de NM, il faut montrer que $\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k)$ est négligeable. Pour cela il faut construire un attaquant $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ sur Π au sens de NM.

$$\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k) = \left\{ \begin{array}{l} (p_k, s_k) \longleftarrow \mathcal{K}(1^k); (m_0, m_1, s) \longleftarrow \mathcal{B}_1^{\mathcal{O}_1}(p_k) \\ b \longleftarrow \{0,1\}; c \longleftarrow \mathcal{E}(p_k, m_b); \widetilde{b} \longleftarrow \mathcal{B}_2^{\mathcal{O}_2}(m_0, m_1, c, s_1) \\ \text{si } b = \widetilde{b}, \text{ retourner 1} \\ \text{sinon retourner 0} \end{array} \right\}$$

et $\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k) = 2\Pr\left[\left(\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k)\right) = 1\right] - 1 \qquad (\alpha_1)$

On dérive l'expression générale suivante de type NM.

$$\mathrm{Exp}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k) = \left\{ \begin{array}{l} (p_k, s_k) \longleftarrow \mathcal{K}(1^k); (\mathcal{M}, s') \longleftarrow \mathcal{A}_1^{\mathcal{O}_1}(p_k) \\ m' \longleftarrow \mathcal{M}; c' \longleftarrow \mathcal{E}(p_k, m'); (R, Y') \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathcal{M}', s', c') \\ X' \longleftarrow \mathcal{D}(Y', p_k); X' \cap Y' = \emptyset \\ m' \notin X' \\ \text{si } R(m', X'), \text{ retourner 1} \\ \text{si non retourner 0} \end{array} \right\}$$

$$\widetilde{\mathrm{Exp}}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k) = \left\{ \begin{array}{l} (p_k, s_k) \longleftarrow \mathcal{K}(1^k); (\mathcal{M}', s') \longleftarrow \mathcal{A}_1^{\mathcal{O}_1}(p_k) \\ m, \widetilde{m} \longleftarrow \mathcal{M}'; c' \longleftarrow \mathcal{E}(p_k, m); (R, Y') \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathcal{M}', s', c') \\ X' \longleftarrow \mathcal{D}(Y', s_k); \\ m \notin X' \\ \text{si } R(\widetilde{m}, X'), \text{ retourner 1} \\ \text{si non retourner 0} \end{array} \right\}$$

Construisons maintenant les événements de ces deux expériences en utilisant $\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{A}, k)$. Définissons alors les événements suivants :

— $\mathcal{A}_1^{\mathcal{O}_1}(p_k)$: $(m_0, m_1, s) \longleftarrow \mathcal{B}_1^{\mathcal{O}_1}(p_k)$, $\mathcal{M}' \longleftarrow \{m_0, m_1\}$ avec la probabilité uniforme ; $s' \longleftarrow (m_0, m_1, p_k, s)$ et retourner $(\mathcal{M}', s')$.

— $\mathcal{A}_2^{\mathcal{O}_2}(\mathcal{M}, s', c')$: avec $s' = (m_0, m_1, p_k, s)$, $\widetilde{b} \longleftarrow \mathcal{B}_2^{\mathcal{O}_2}(m_0, m_1, s, c)$, $c' \longleftarrow \mathcal{E}(p_k, \overline{m_{\widetilde{b}}})$.

— Définir $R(u, v)$, si et seulement si $u = \overline{v}$, poser $Y' = \{c'\}$, retourner (R, Y').

— On pose $X' = \{\overline{m_{\widetilde{b}}}\}$, donc on a bien $X' \longleftarrow \mathcal{D}(Y', s_k)$, et $m_b \notin X$. D'après la proposition précédente, on peut considérer que $m_0 \neq m_1$ car l'avantage de l'attaquant ne change pas si $m_0 = m_1$.

— On sait que :

$\mathrm{Adv}_{\Pi}^{\mathrm{NM}} = \left|\Pr\left[(\mathrm{Exp}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)) = 1\right] - \Pr\left[(\widetilde{\mathrm{Exp}}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)) = 1\right]\right|$

De plus on a le résultat suivant :

$\Pr\left[(\mathrm{Exp}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)) = 1\right] = \Pr\left[(\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k)) = 1\right] \qquad (\alpha_2)$

car $R(m_b, X') = R(m_b, \overline{m_{\widetilde{b}}}) \Longleftrightarrow m_b = \overline{\overline{m}_{\widetilde{b}}} \Longleftrightarrow m_b = m_{\widetilde{b}} \Longleftrightarrow b = \widetilde{b}$ puisque $m_0 \neq m_1$ d'après l'hypothèse précédente ; donc $\mathrm{Exp}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)) = 1 \Longleftrightarrow \mathrm{Exp}_{\Pi}^{\mathrm{NM}}(\mathcal{B}, k)) = 1$.

— On a aussi $\Pr\left[(\widetilde{\mathrm{Exp}}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)) = 1\right] = 1/2 \qquad (\alpha_3)$

car $\mathcal{A}_2$ n'a aucune information sur m, donc s'il produit X', il a autant de chances d'avoir $R(\widetilde{m}, X')$ que non$R(\widetilde{m}, X')$.

— Maintenant d'après $\alpha_1, \alpha_2, \alpha_3$, on a

$\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k) = 2\Pr\left[\left(\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k)\right) = 1\right] - 1$

$= 2\left[\Pr\left[\left(\mathrm{Exp}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k)\right) = 1\right] - 1/2\right]$ et d'après α_1 et α_2

$= 2\left[\Pr\left[\left(\mathrm{Exp}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)\right) = 1\right] - \Pr\left[\left(\widetilde{\mathrm{Exp}}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)\right) = 1\right]\right]$

$= 2\mathrm{Adv}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)$

Ainsi $\mathrm{Adv}_{\Pi}^{\mathrm{IND}}(\mathcal{B}, k)$ est négligeable si $\mathrm{Adv}_{\Pi}^{\mathrm{NM}}(\mathcal{A}, k)$ l'est.

Chapitre 7

Objectifs de sécurité et Formalisme de l'attaquant sur les signatures

7.1 Objectifs de sécurité pour les signatures

7.1.1 Signature numérique

: Nous reprenons ici la définition avec un paramètre de sécurité.

Une signature aléatoire est un triplet $(Genkey,\ Sig,\ Ver)$

- **Genkey :** prend en entrée un paramètre de sécurité k et sort un couple de clés privée/publique (p_k, s_k).

- **Sig :** algorithme de signature :

- prend en entrée un paramètre de sécurité k et fabrique une valeur aléatoire σ.

- prend en entrée (s_k, m, ρ) où m est un texte à signer et produit une signature σ. [NB : il y a des variantes pour lesquelles ρ n'est pas donné en sortie].

- **Ver :** algorithme de vérification :

prend en entrée (m, ρ, σ, p_k) ; l'algorithme de vérification sort 1 si la signature est avlide et 0 sinon.

7.1.2 Objectifs de sécurité des signatures

Pour une signature, l'attaquant cherche principalement à falsifier une signature ou à créer un message et sa signature sans connaitre la clé privée.

1. **Unbreakability (UB) :** l'attaquant retrouve la clé privée (ou clé équivalente) à partir de la clé publique (cas de RSA).
2. **Existential Unforgeability (EUF) :** l'attaquant crée un message et une signature valide.

7.1.3 Le modèle de l'oracle aléatoire pour les signatures

Concernant les signatures, pour une preuve dans le Modèle de l'oracle aléatoire, (voir [1] et[6]) :

- une fonction de hachage est utilisée comme une fonction aléatoire dans le processus de simulation (elle est l'oracle de hachage) ;
- le seul moyen de calculer la fonction de hachage est d'invoquer l'oracle de hachage ;
- L'algorithme de simulation $\mathcal{S}$ doit simuler l'environnement de l'attaquant $\mathcal{A}$ avec une information publique seulement ;
- à la fin de la preuve, si l'attaquant $\mathcal{A}$ réussit dans quelques attaques (falsifie une signature ou crée un message et sa signature sans connaitre la clé privée) alors $\mathcal{S}$ peut être en mesure de résoudre le problème difficile connu de la notion de sécurité. dans la preuve on cache l'instance difficile du problème à résoudre dans dans la fonction de hachage (mais il faut que les réponses à l'attaquant restent parfaitement aléatoire) et éventuellement dans la clé publique lors de sa fabrication.
-

7.2 Formalisme de l'attaquant sur les signatures

Goldwasser, Micali and Rivest (in 1988) dans[8], on introduit le concept fondamental de de sécurité sur les signatures appelé : "Existential unforgeability with respect to adaptive chosen- message attacks".

Pour cela, une réduction algorithm $\mathcal{R}$ et un attaquant $\mathcal{A}$, simule a le jeu suivant .

Démarrage : $\mathcal{R}$ déroule l'algorithme Genkey un paramètre sécurité k en entrée, pour obtenir une clé publique p_{key} et une clé secrète s_{key}, et envoiep_{key} à l'adversaire.

Requêtes : En procédant de façon adaptative, $\mathcal{A}$ peut demander une signature sur n'importe qu'elle message $m \in \mathcal{M}$ (plusieurs signatures d'un même message peuvent être demandées grâce au caractère aléatoire de la fonction de signature) et $\mathcal{R}$ doit répondre avec (m, r, σ) où $\sigma = Sig(s_{key}, m, r)$ et r une valeur aléatoire.

Soit $Hist(\mathcal{S})$ une base de données des signatures déjà calculer par l'oracle de signature suite aux sollicitations de $\mathcal{A}$).

Sortie : Éventuellement, $\mathcal{A}$ va sortir une pair (m, r, σ) et dans ce cas gane le jeu si $Ver(p_{key}, m, r, \sigma) = 1$ et si $(m, r, \sigma) \notin Hist(\mathcal{S})$ (cette dernière condition oblige l'attaquant $\mathcal{A}$ à forger sa propre signature).

Unforgeability against Adaptative Chosen Message Attack (EUF-CMA) : l'attaquant est autorisé à avoir accés à un oracle de signature (sans limite) et peut solliciter la signature de n'importe quel message (plusieurs sollicitations du même message sont possibles grâce à la valeur aléatoire).

Chosen Message Security :

EUF-CMA est le niveau supérieur de sécurité pour les signatures (l'équivalent de CCA2 pour le chiffrement).

Une signature est dite (q, ε, τ)-sûre si pour tout attaquant $\mathcal{A}$dont le temps de travail est majoré par τ, on ait :

$$\mathrm{Succ}^{\mathrm{EUF-CMA}}(\mathcal{A}, \mathcal{P}) = \left\{ \begin{array}{l} (p_k, s_k) \longleftarrow \mathrm{Genkey}(1^k), \\ (m^*, \rho^*, \sigma^*) \longleftarrow \mathcal{A}(p_k), \\ \mathrm{Ver}(p_k, \mathcal{M}^*, \rho^*, \sigma^*) = 1, \\ (m^*, \rho^*) \notin Hist(S) \end{array} \right\}$$

où :

- **Genkey** est l'algorithme de génération de clés ;

- $\mathcal{A}^{\mathbf{Sig}(S_k,.)}(p_k)$ signifie que l'attaquant peut prendre en entrée la clé publique et solliciter l'oracle de signature $\mathrm{Sig}(S_k, .)$.

- **Hist(S)** est l'ensemble des couples $m, \rho)$ pour lesquels l'oracle $\mathrm{Sig}(S_k, .)$ a produit une signature.

Chapitre 8

Exemples de preuves de sécurité

8.1 Signature de Groth (One Time Signature) dans le modèle standard

Groth : Simulation-sound NIZK proofs for a practical language and constant size group signatures. Advances in Cryptology - Asiacrypt 2006 : pp. 444-459

One Time Signature

— G un groupe, $g \in G$ avec $\#\langle g \rangle = p$ premier

— GenKey : $p_k = (X = g^x, Y = g^y, Z = g^z)$ clé publique.

— Sig : $m \in (\mathbb{Z}/p\mathbb{Z})^*,\ r \longleftarrow (\mathbb{Z}/p\mathbb{Z})^*$

Calculer $s = (1 - mx - yr)/z\ \in (\mathbb{Z}/p\mathbb{Z})^*$, retourner $\sigma = (r, s)$.

— Ver : $\sigma = (r, s)$ sur m

Calculer $X^m Y^r Z^s \stackrel{?}{=} g$

Théorème 8.1.1. *Si le logarithme discret est difficile dans G alors la signature de Groth est sûre One Time EUF-CMA dans le modèle standard.*

Preuve :

- On note $(g, h = g^\alpha)$ l'instance difficile à casser.

- On le cache dans la clé publique en prenant $X = g^{a_1} h^{b_1},\ Y = g^{a_2} h^{b_2}$ *et* $Z = g^{a_3}$ où $a_1, b_1, a_2, b_2, a_3 \stackrel{Rand}{\longleftarrow} (\mathbb{Z}/p\mathbb{Z})^*$

- le propriétaire de la clé publique gère l'oracle de signature donc doit être capable de signer : si on donne m, on calcule $r = -mb_1/b_2 \mod$ et $s = (1 - ma_1 - ra_2)/a_3 \mod p$.

- Si l'attaquant fabrique une signature $\sigma_0 = (r_0, s_0)$ pour un texte m_0, alors

$X^{m_0}Y^{r_0}Z^{s_0} = g \iff \alpha(b_1m_0 + b_2r_0) = 1 - m_0a_1 - s_0a_3 - r_0a_2 \implies \alpha = (1 - m_0a_1 - s_0a_3)/(b_1m_0 + b_2r_0)$ donc l'instance du logarithme discret est résolu (évidemment avec une forte probabilité $(b_1m_0 + b_2r_0) \neq 0$ car l'attaquant ne connait pas b_1 et b_2.

8.2 Signature RSA-PSS-R avec un exposent aléatoire dans le modèle oracle aléatoire

Noter que ce schémas de signature n'a pas été publié.

Définition 8.2.1. ***RSA problem.***

Soit $n = pqun$ module RSA, e un entier co-premier $\varphi(n) = (p-1)(q-1)$ and $z \in (\frac{\mathbb{Z}}{n\mathbb{Z}})^*$. Trouver x tel que : $x^e = z \, modn$.

Un algorithme $\mathcal{R}$ est dit $(\tau_{\mathcal{R}}, \varepsilon_{\mathcal{R}})$-résoudre la problème RSA, si en au plus $\tau_{\mathcal{R}}$ opérations, $\Pr\{(n,e) \leftarrow RSA(1^k), z \leftarrow (\frac{\mathbb{Z}}{n\mathbb{Z}})^*, x \leftarrow \mathcal{R}(n,e,z), x^e = z \mod n\} \geq \varepsilon_{\mathcal{R}}$*, où les probabilités sont calculés sur les distributions de (n,z) et sur les valeurs aléatoires de $\mathcal{R}$.*

Dans cette sous-section, nous présentons une signature appelé RSA-PSS-R similaire à PSS mais avec un exposant aléatoire.

Procédé de signature .

Génération des clés $G_{key}(1^k)$

1. Choir aléatoirement p, q et calculer $n = pq \leftarrow RSA(1^k)$ où $p = 2p_1^{\beta_1} + 1$ et $q = 2p_2^{\beta_2} + 1$
2. Poser $p_k = (n)$ et $s_{key} = (p, q)$.
3. Soient G, H et T des fonctions de hachage cryptographie, telles que T sort des entiers impairs, avec $T : \{0,1\}^{k_r} \to \{0,1\}^{k_e} \bigcap \mathbb{N}_{odd}$, $\quad H : \{0,1\}^{k_r + k_m + k_e} \to \{0,1\}^{k_\omega}$ and $G : \{0,1\}^{k_\omega + k_e} \to \{0,1\}^{k_r + k_m}$

Nota bene 1 : Pour construire T, on peut procéder comme suit :

Soit $T' : \{0,1\}^* \to \{0,1\}^{k_1}$ une fonction de hachage, k_1 un paramètre de sécurité, on définit alors T par $T(x) = 1||T'(x)||1 \in [0, n-1] \cap [2^{k_1+1}, 2^{k_1+2}[\cap \mathbb{N}_{odd}$ où $\mathbb{N}_{odd}$ est l'ensemble des entiers impaires.

Nota bene 2 :

Comme $T(x)$ est impaire alors avec une très grand probabilité, $T(x)$ est co-premier avec $\varphi(n) = 4p_1^{\beta_1}p_2^{\beta_2}$, trouver une entier impair qui divise $\varphi(n)$, revient à factoriser n (ce est supposé comme étant difficile pour des paramètres adéquats).

Algorithme de signature $S(s_{key}, m)$

1. Sélectionner aléatoirement r et calculer $e = T(r)$ et $d = e^{-1} mod \varphi(n)$
2. calculer $\omega = H(r||m||e)$ and $s = (r||m) \oplus G(\omega||e)$
3. Poser $\mu(r, m) = 0||\omega||s$
4. calculer $\sigma = \mu(r, m)^d mod n$ et sortir $\big(e, \sigma\big)$

Algorithme de vérification $V(p_{key},)$ (m n'est pas donné)

1. Calculer $b||\omega||s = \sigma^e mod n$
2. si $b = 1$ sortir 0
3. calculer $r||m = s \oplus G(\omega||e)$
4. si $T(r) = e$ et $H(r||m||e) = \omega$ sortir 1 (et m) si non sortir 0

Preuve de sécurité : Réduction dans la modèle de l'oracle

Théorème 8.2.1. *S'il existe un attaquant $\mathcal{A}$ qui $(q_T, q_H, q_G, q_{sig}, \tau_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-résout EUF-BR-CMA(RSA-PSS-R-RE), alors il existe un algorithme de réduction $\mathcal{R}$ simulant l'environnement de $\mathcal{A}$ dans le modèle de l'oracle aléatoire qui $(\tau_{\mathcal{R}}, \varepsilon_{\mathcal{R}})$-résout le problème RSA avec une probabilité de succès de $\varepsilon_{\mathcal{R}} = \varepsilon_{\mathcal{A}}(1 - \frac{1}{2^{|r|}})$ et un temps borné par $\tau_{\mathcal{R}} \leq \tau_{\mathcal{A}} + q_T\mathcal{O}(1) + q_G(q_H + q_T)\mathcal{O}(1) + (q_H + q_{sig} + 2)\mathcal{O}(k^3)$ où $J = 2^{|r|}$ est le nombre de valeurs aléatoires utilisables dans exponentiation, $\mathcal{R}$ reçoit de $\mathcal{A}$, q_{sig} requêtes de signature, q_H, q_G et q_T requêtes pour les oracles de hachage H , G et T respectivement, et k est un paramètre de sécurité.*

proof

Notre réduction $\mathcal{R}$ fonctionne comme suit.

1. Un paramètre de sécurité k, un attaquant $\mathcal{A}$ qui $(q_T, q_H, q_G, q_T, q_{sig}, \tau_{\mathcal{A}}, \varepsilon_{\mathcal{A}})$-résout EUF-BR- CMA(RSA-PSS-R-RE) et un algorithme simulateur $\mathcal{R}$ sont donnés.
2. $\mathcal{R}$ simule G_{key} et transmet une clés publique $pk = n$ à $\mathcal{A}$,
3. $\mathcal{R}$ reçoit des requêtes pour T de $\mathcal{A}$: il simule T au plus q_T fois ,
4. $\mathcal{R}$ reçoit des requêtes pour$G$ de $\mathcal{A}$: il simule$G$ au plus q_G times,
5. $\mathcal{R}$ reçoit des requêtes pour H de $\mathcal{A}$: il simule H au plus q_H times,
6. $\mathcal{R}$ reçoit des requêtes de signatures de $\mathcal{A}$: il simule un oracle de signature au plus q_{sig} fois,
7. $\mathcal{A}$ sort une signature forgée (e^*, σ_*) pour RSA-PSS-R-RE,
8. $\mathcal{R}$ simule une vérification de la signature forgée qui sera valide avec une probabilité $\varepsilon_{\mathcal{A}}$,
9. $\mathcal{R}$ sort x tel que $x^v = y \; mod n$.

Simulation de l'oracle de génération clé $\mathbf{G}_{key}$; l'algorithme de réduction $\mathcal{R}$

— pose $Hist(S) = \emptyset$ (Oracle de signature base de données)
— pose $Hist[T] = \emptyset$ (Oracle **T** base de données)
— pose $Hist[G] = \emptyset$ (Oracle **G** base de données)
— pose $Hist[H] = \emptyset$ (Oracle **H** base de données)
— envoie la clé publique n à $\mathcal{A}$.
— sélectionne N (avec $N < 2^{|m|}$) valeurs aléatoires $i_1, ..., i_N \in [1, 2^{|m|}]$ où $2^{|m|}$ est le nombre de message à signer ; et pose $Z = \{i_1, ..., i_N\}$.

Simulation de l'oracle de hachage T ; quand $\mathcal{A}$ envoie une requête à T avec un message m_j, $1 \leq j \leq 2^{|m|}$

— $\mathcal{R}$ regarde dans $Hist[T]$, si m_j a été envoyé dans le passé pour une signature. Si $T(m_j)$, est déjà défini comme une valeur T_{m_j}, $\mathcal{R}$ retourne cette valeur,
— si $j \notin Z$ $\mathcal{R}$ choisit aléatoirement $t_{m_j} \leftarrow [0, n-1] \cap \mathbb{N}_{odd}$ et définit $T(m_j) = t_{m_j}$,
— si $j \in Z$: $\mathcal{R}$ choisit aléatoirement $t'_{m_j} \leftarrow [0, n-1] \cap \mathbb{N}_{odd}$ et définit $T(m_j) = vt'_{m_j}$,
— $\mathcal{R}$ mémorise $\big(m_j, T(m_j)\big)$ in $Hist[T]$.

Simulation de l'oracle hachage H ; quand $\mathcal{A}$ sollicite H avec un message et une valeur aléatoire (m, r_j),

— $\mathcal{R}$ utilise sa propre simulation pour calculer $e_j = T(r_j)$,

— $\mathcal{R}$ regarde dans $Hist[H]$, si (m, r_j) a été envoyé à cet oracle dans le passé. Si $H(r_j||m||e_j)$ est déjà défini comme $h_{r_j||m||e_j}$, $\mathcal{R}$ retourne cette valeur $h_{r_j||m||e_j}$;

— if $j \notin Z$, $\mathcal{R}$;

— choisit aléatoirement $u_{j,m}$ tel que $(u_{j,m})^{e_j} mod n = 0||\omega_{j,m}||s_{j,m}$;

— définit et retourne $H(r_j||m||e_j) = \omega_{j,m}$ to $\mathcal{A}$;

— mémorise $(m, r_{j,m}, e_j, s_{j,m}, u_{j,m}, \omega_{j,m})$ dans $Hist[H]$

— si $j \in Z$, $\mathcal{R}$;

— choisit $u_{j;m}$ tel que $y(u_{j,m})^{e_j} mod n = 0||\omega_{j,m}||s_{j,m}$;

— définit et retourne $H(r_j||m||e_j) = \omega_{j,m}$ à $\mathcal{A}$;

— mémorise $(m, r_j, e_j, s_{j,m}, \perp, y)$ dans $Hist[H]$;

Simulation de l'oracle de hachage G Quand $\mathcal{A}$ envoie une nouvelle requête (e, ω) ;

— $\mathcal{R}$ regarde dans $Hist[T]$, si e a été calculer dans le passé, si oui retourne cette valeur.

— si non $\mathcal{R}$ choisit aléatoirement g_0, définit puis retourne $G(\omega||e) = g_0$ to $\mathcal{A}$;

— $\mathcal{R}$ rend l'unique $(r_j, e_j) \in Hist[T]$ tel que $e = e_j$

— $\mathcal{R}$ regarde dans $Hist[H]$, s'il existe $(m, r_j, e_j, \alpha, \beta, \gamma)$ tel que $\gamma = \omega$

— Si non, $\mathcal{R}$ choisit aléatoirement g_1, définit et retourne $G(\omega||e) = g_1$ à $\mathcal{A}$;

— $\mathcal{R}$ prend l'unique $(m, r_j, e_j, \alpha, \beta, \omega)$ in $Hist[H]$, définit et retourne $G(\omega||e) = (r_j||m) \oplus \alpha$ to $\mathcal{A}$;

— $\mathcal{R}$ mémorise $\big(r_j, e_j, \omega, (r_j||m) \oplus \alpha\big)$ dans $Hist[G]$;

Simulation de l'oracle de signature $\mathbf{S^{T,H,G}}$; Quand $\mathcal{A}$ sollicite une signature pour un message m, $\mathcal{R}$.

sélectionne aléatoirement j in $[1, 2^{|r|}] \setminus Z$, puis $\mathcal{R}$

— fait appel à sa propre simulation de T, H et G pour calculer $e_j = T(r_j)$, $H(r_j||m||e_j)$

— cherche l'unique $(m, r_j, e_j, \alpha, \beta, \gamma)$ dans $Hist[H]$ et retourne (e_j, β) ;

— garde (e_j, β) dans la base de données $Hist[S]$ de l'oracle de signature.

Simulation de la vérification $\mathbf{V^{T,H,G}}$; Étant donnée une signature (e, σ), $\mathcal{R}$

— calcule $b||\omega||s = \sigma^e mod n$;

— si $b = 1$ output 0 ;

— fait appel à la simulation de G pour obtenir $G(\omega||e)$;

— pose $r||m = s \oplus G(\omega||e)$;

— fait appel aux simulations de T et Hpour obtenir $T(r)$ and $H(r||m||e)$;

— si $e = T(r)$ et $H(r||m||e) = \omega$ sort 1 (et m) si non sort 0.

Synthèse : On suppose qu'à la fin du jeu, $\mathcal{A}$ sort (e^*, σ_*) comme une signature forgée. Alors,

— $\mathcal{R}$ simule $V^{T,H,G}$ pour vérifier si (e^*, σ_*) est une signature valide

— si la vérification retourne 0 ou $(e^*, \sigma_*) \in Hist(S)$, $\mathcal{R}$ arrête le jeu ;

— $\mathcal{R}$ calcule $\sigma_*^{e^*} mod n = 0||\omega||s$;

— $\mathcal{R}$ pose $r||m = s \oplus G(\omega||e^*)$;

— $\mathcal{R}$ cherche dans $(m, r, e^*, s, u, \omega) \in Hist[H]$; il existe $j \in [1, 2^{|r|}]$ tel que $r = r_{j_0}$ et $e^* = e_{j_0}$;

— si $j_0 \notin Z$, $\mathcal{R}$ arrête le jeu ;

— $\mathcal{R}$ pose $x = \left(\frac{\sigma_*}{u}\right)^{e_{j_0}/v}$

— $\mathcal{R}$ sort x

Qualité de la réduction :

— $\mathcal{R}$ simule parfaitement le schémas de signature (oracles de hachage, génération de clés, signature and vérification) avec une probabilité 1.

— $\mathcal{A}$ sort alors (e^*, σ^*) avec une probabilité d'au moins $\varepsilon_{\mathcal{A}}$ durant un temps $\tau_{\mathcal{A}}$,

— Comme Z ne dépend pas de $\mathcal{A}$, l'événement $j_0 \in Z$ apparait avec une probabilité de $\frac{N}{2^{|r|}}$.

— Ainsi $\mathcal{R}$ trouve une solution x telle que $x^v = y \ mod n$, avec une probabilité 1.

En calculant tout ça, $\mathcal{R}$ réussit à résoudre RSA avec une probabilité $\varepsilon_{\mathcal{R}} = \frac{N}{2^{|r|}}.\varepsilon_{\mathcal{A}}$

en un temps $\tau_{\mathcal{R}}$ majoré par $\tau_{\mathcal{R}} \leq \tau_{\mathcal{A}} + q_T\mathcal{O}(1) + q_G(q_H + q_T)\mathcal{O}(1) + (q_H + q_{sig} + 2)\mathcal{O}(k^3)$ où N est le nombre de valeurs aléatoires autorisés pour RSA-PSS-R-RE signature.

Pour avoir $\varepsilon_{\mathcal{R}} \geq \varepsilon_{\mathcal{A}}(1 - \frac{1}{2^{|r|}})$ il suffit de choisir $N = 2^{|r|} - 1$

□

Nota Bene : Noter que q_{sig}, q_G, q_T et q_H ne modifie pas la probabilité de succès mais uniquement le temps de réduction sous forme polynomiale.

8.3 GHR Signature dans le modèle standard

J.S. Coron,*Optimal Security proofs for PSS and other signature schemes*, Proceedings of Eurocrypt'02, lncs, vol. 2332, Springer-Verlag, 2002, pp. 272-287.

GHR Signature

- GenKey : $n = pq$, $p = 2p' + 1$, $p = 2p' + 1$; $S \in (\mathbb{Z}/n\mathbb{Z})^*$, s aléatoire.

$\psi : \{0,1\} \longrightarrow Premiers \geq 3,\ (l = 3)$

(n, s) clé publique, (p, q) clé privée.

- Sig : $m \in \{0,1\}$, on calcule $\sigma = s^{1/\psi(m)} \mod n$.

- Ver : de σ sur m.

$\sigma^{\psi(m)} = s \mod n$

il faut que $\psi(m)$ soit inversible donc en fait un nombre premier $\neq q$ *et* p'

Théorème 8.3.1. *Si Strong RSA est difficile, GHR est EUF-CMA sûr dans le modèle standard.*

Preuve : A faire.

8.4 Chiffrement El Gamal en modèle standard

8.4.1 Problèmes DLP, CDH, DDH

Soit G un groupe admettant un sous groupe cyclique H d'ordre q assez grand et g un générateur de $H = \langle g \rangle$.

- Le Problème du Logarithme Discret (DLP)

Etant donné $y \in H$, calculer x tel que $y = g^x$. On pose alors $x = \log_g y$. Le succès d'un attaquant $\mathcal{A}$ est donné par $Succ^{DLP}(\mathcal{A}) = Pr[\mathcal{A}(g^x) = x]$.

- Le Problème Calculatoire de Diffie-Hellman (CDH)

Étant donné deux éléments $\alpha,\ \beta \in H,\ \alpha = g^a,\ \beta = g^b,\ calculer\ \gamma = g^{ab}$.

On définit $\gamma = CDH(\alpha, \beta)$. $Succ^{CDH}(\mathcal{A}) = Pr_{x \in \mathbb{Z}/q\mathbb{Z}}[\mathcal{A}(g^a, g^b) = g^{ab}]$

- Le Problème Décisionnel de Diffie-Hellman (DDH)

Étant donnés trois éléments $\alpha,\ \beta,\ \gamma \in H$. Décider si $\gamma = CDH(\alpha, \beta)$ i.e. aussi si $\alpha = g^a,\ \beta = g^b,\ \gamma^c$, décider si $c = ab \mod q$.

L'algorithme qui résout DDH est appelé **distingueur** $\mathcal{D}$.

L'avantage est donné par :

$Adv^{DDH}(\mathcal{D}) = \left| Pr_{a,b,c \in \mathbb{Z}/q\mathbb{Z}} \left[1 \longleftarrow \mathcal{D}(g^a, g^b, g^c)\right] - Pr_{a,b \in \mathbb{Z}/q\mathbb{Z}} \left[1 \longleftarrow \mathcal{D}(g^a, g^b, g^{ab})\right] \right|$

8.4.2 Système El Gamal sur les corps premiers :

Soient p et q deux nombres premiers tel que q divise $p-1$. Soit g un générateur de $(\mathbb{Z}/q\mathbb{Z})^*$.

Clé privée de Bob : (x, p, q, g), $x \xleftarrow{Rand} \mathbb{Z}/q\mathbb{Z}$

Clé publique de Bob : (y, p, q, g), $y = g^x \mod p$.

Chiffrement : Alice choisit $m \in \mathbb{Z}/q\mathbb{Z}$ puis $k \xleftarrow{Rand} \mathbb{Z}/q\mathbb{Z}$ et calcule $m_1 = g^k \mod p$ et $m_2 = y^k m \mod p$

Le chiffré $c = (m_1, m_2)$.

Déchiffrement : $m = m_2 m_1^{-x} \mod p$

Théorème 8.4.1. *$CDH \iff$ OW-CPA.*

L'inversion (OW-CPA) du chiffrement El Gamal est équivalente au Problème Diffie-Hellman Calculatoire c'est à dire pour un paramètre de sécurité t, s'il existe un attaquant $\mathcal{A}$ qui inverse El Gamal alors on peut construire un algorithme $\mathcal{B}$ qui résout CDH,

$\left(Succ^{OW-CPA}(\mathcal{A}, t) \leq Succ^{CDH}(\mathcal{B}, t)\right)$ *:*

Preuve :

Soit une instance aléatoire $(\alpha = g^a, \beta = g^b)$ du problème CDH que l'on veut résoudre, considérant l'attaquant $\mathcal{A}$ contre OW-CPA. On note Exp_0 le scénarios de l'attaquant, on le modifie progressivement pour résoudre CDH. On note C le challenger.

$Exp_0(\mathcal{A}, k)$

$x \xleftarrow{Rand} \mathbb{Z}/q\mathbb{Z}\ : C$

$y = g^x$

$(p_k, s_k) = (y, x)$

$m \overset{Rand}{\longleftarrow} H \ : C$

$k \longleftarrow \mathbb{Z}/q\mathbb{Z}$

$m_1 = g^k \mod p,\ m_2 = y^k m_2 \mod p$

$\widetilde{m} \longleftarrow \mathcal{A}(y, m_1, m_2)$

On note $S_0 =$ "$m = \widetilde{m}$" et $\Pr(S_0) = \varepsilon$

On modélise le jeu réel de façon successive :

$\underline{\mathrm{Exp}_1(\mathcal{A}, k)}$

Dans Exp_0, on remplace $(x, y = g^x)$ par $(a, \alpha = g_a)$ cela est possible car $(\alpha = g^a, \beta = g^b)$ est aléatoire.

On note $S_1 =$ "$m = \widetilde{m}$" et on a $\Pr(S_1) = \Pr(S_0)$.

$\underline{\mathrm{Exp}_2(\mathcal{A}, k)}$

Dans Exp_1 , on remplace $m_1 = g^k$ par $m_1 = g^b = \beta$ et on note $S_2 =$ "$m = \widetilde{m}$" alors $\Pr(S_2) = \Pr(S_1)$ car β est aléatoire.

$\underline{\mathrm{Exp}_3(\mathcal{A}, k)}$

Dans Exp_2, on remplace $m_2 = my^k$ où $m \overset{Rand}{\longleftarrow} H$ par $m_2 \overset{Rand}{\longleftarrow} \mathbb{Z}/q\mathbb{Z}$ (aléatoire, créer directement)

La structure de groupe fait que la distribution est uniforme pour les 2 facteurs de m_2 car m et y^k sont dans le même sous-groupe cyclique H, on note $S_3 =$ "$Sm_1^{-x} = m$"

$\Pr(S_3) = \Pr(S_1)$.

On a maintenant

$$\varepsilon = Pr(S_0) = Pr(S_3)$$

$$= \Pr\left[\mathcal{A}(y, m_1, m_2) = m = S/m_1^x : \left\{ \begin{array}{l} S \overset{Rand}{\longleftarrow} H,\ y = g^a \\ a, b \longleftarrow \mathbb{Z}/q\mathbb{Z},\ m_1 = g^b \end{array} \right.\right]$$

$$= Pr\left[\mathcal{A}(y, m_1, m_2) = S/CHD(\alpha, \beta) : \left\{ \begin{array}{l} \alpha, \beta, S \longleftarrow H \\ y = \alpha,\ m_1 = \beta \end{array} \right.\right]$$

car si $m = S/m_1^x$ alors $g^{ab} = S/m$.

Théorème 8.4.2. *La sécurité sémantique IND-CPA du chiffrement El Gamal est équivalente au problème Diffie-Hellman Décisionnel.* $\mathrm{Adv}^{\mathrm{IND-CPA}}(\mathcal{A}, t) \leq 2 \times \mathrm{Adv}^{\mathrm{DDH}}(\mathcal{D}, t)$ *où* $\mathcal{A}$ *est un*

attaquant El Gamal et $\mathcal{D}$ distingueur Diffie-Hellman.

Preuve :

Soit $(\alpha = g^a, \beta = g^b)$ une instance aléatoire. Notons $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ un attaquant contre le système El Gamal en temps t.

$\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

Exp_0 :

$(y = g^x, x \overset{Rand}{\longleftarrow} \mathbb{Z}/q\mathbb{Z}) \longleftarrow \mathcal{K}(1^k) \ : C$

$(m_0, m_1 \longleftarrow \mathcal{A}_1^{\mathcal{D}}(y)$

$(\widetilde{m}_1, \widetilde{m}_2) \longleftarrow \mathcal{E}(y, m_\delta, k^\delta) \overset{Rand}{\longleftarrow} \{0,1\} \ : C$

$\{0,1\} \ni \delta' \longleftarrow \mathcal{A}_2^{\mathcal{D}}(\widetilde{m}_1, \widetilde{m}_2)$

On note $\varepsilon' = \Pr(\delta = \delta') = \Pr(S_0)$.

Exp_1 :

On modifie $(y, \widetilde{m}_1)$ en (α, β), $\widetilde{m}_2$ reste inchangé (donc $x = a, \ h = b$).

Les distributions de $x, \ y$ sont identiques car (α, β, a) sont aléatoires.

On note S_1 l'événement associé alors $\Pr(S_1) = \Pr(S_0)$.

Exp_2 :

On modifie $\widetilde{m}_2$ dans le cas précédent au lieu de $\widetilde{m}_2 = m\gamma y^k$ on pose $\widetilde{m}_2 = m\delta C$ où $C = \text{DH}(A, B)$. Si S_2 est l'événement aléatoire associé alors $\Pr(S_2) = \Pr(S_1)$.

Exp_3 :

Ici on remplace $C = \text{DH}(A, B)$ par $C = g^c$, c aléatoire.

On note S_3 l'événement associé.

Maintenant, étant donné que $\gamma' = \gamma$ est un événement détectable, on peut définir un algorithme distingueur qui :

- exécute le jeu Exp_2 si $C = \text{DH}(A, B)$
- exécute le jeu Exp_2 si $C \neq \text{DH}(A, B)$

et qui

- retourne 1 si $\gamma' = \gamma$
- retourne 0 si $\gamma' \neq \gamma$.

Mais, on a :

$\Pr[1 \longleftarrow \mathcal{D}/C \overset{Rand}{\longleftarrow}] = \Pr[S_2]$

$\Pr[1 \longleftarrow \mathcal{D}/C = \mathrm{DH}(A, B)] = \Pr[S_3]$

Mais considérant que $\mathrm{DH}(A, B)$ et C pour Exp_3 sont tous les deux parfaitement aléatoires, on voit que $Pr[S_3] = 1/2$

En posant $\varepsilon' = \varepsilon/2$, on a $\mathrm{Adv}^{DDH}(t) \geq \Pr[S_2] - \Pr[S_3] = \frac{1+\varepsilon}{2} - \frac{1}{2} = \varepsilon$

par suite $\varepsilon \leq 2\mathrm{Adv}^{DDH}(t)$ d'où l'avantage recherché.

□

NB : (très important)

il n'y a pas de non malléabilité pour le chiffrement El Gamal à cause de la propriété d'homomorphisme.

Bibliographie

[1] M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Communications Security, ACM, 1993.

[2] M. Bellare and P. Rogaway, *The Exact security of digital signatures : How to sign with rsa and Rabin*, Proceedings of Eurocrypt 1996, lncs, vol. 1070, Springer-Verlag, 1996, pp. 399-416.

[3] P. Barthelemy, R. Rolland P. Veron *Cryptographie - principes et mises en oeuvre*, Lavoisier, 2005.

[4] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway *"Rela- tions among notions of security for public-key encryption schemes"*, in Advances in Cryptology-Crypto ?98, Lecture Notes in Computer Science, vol. 1462, Springer-Verlag, 1998, p. 26-45.

[5] M. Bellare and A. Sahai *"Non-malleable encryption : equivalence bet- ween two notions, and an indistinguishability-based characterization"*, in Advances in Cryptology-Crypto 99, Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1999, p. 519-536.

[6] R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology*, revisited, STOC 98, ACM, 1998.

[7] R. Cramer and V. Shoup *"A practical public key cryptosystem pro- vably secure against adaptative chosen ciphertext attack"*, in Advances

[8] . S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2), pp. 281-308, April 1988.

[9] M.R. Garey and D.S. Johnson :"Computers and Intractability : A Guide to the Theory of NP-Completeness" New York : W. H. Freeman. 1983. ISBN 0-7167-1045-5.

[10] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[11] D. Pointcheval *Provable Security for Public Key Schemes* Advanced Course on Contemporary Cryptology, pages 133-189, June 2005. http ://www.di.ens.fr/ pointche/pub.php ?reference=Po04

[12] D. Pointcheval and J. Stern *Security Proofs for Signature Schemes* Advances in Cryptology Proceedings of EUROCRYPT '96 (may 12-16, 1996, Zaragoza, Spain) U. Maurer, Ed. Springer-Verlag, LNCS 1070, pages 387-398.

[13] . R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978

[14] R. Rolland *Cours de sécurité prouvé*, 2008.

[15] Jonathan Katz, Yehuda Lindell *Introduction to Modern Cryptography*

[16] D. Vergnaud *Course "Provable Security in Public-Key Cryptography"* Lecture 1, 2, 3 and 4 in June 2010 at Dakar available at http ://www.di.ens.fr/ damien (Ecole Normale Superieure, C.N.R.S. I.N.R.I.A. (France))