Cours de cryptologie avancée

Guilhem Castagnos

Septembre – Décembre 2023

version du 3 septembre 2023

Table des matières

I Bibliographie **1**

II Sécurité CPA **3**
1 Introduction à la sécurité réductionniste 3
2 Fonctions à sens unique et à trappe 4
3 Schéma de chiffrement asymétrique 6
4 Sécurité sémantique 7
5 Les preuves par jeux 14
6 Non malléabilité 15
7 Résumé 15

III Quelques applications **17**
1 Chiffrement linéairement homomorphe 17
2 Vote électronique 18
3 Partage de secret 18
4 Chiffrement à seuil 20

Chapitre I

Bibliographie

Jonathan Katz and Yehuda Lindell *Introduction to Modern Cryptography* 2007 by Chapman & Hall/CRC Press

Oded Goldreich *Foundations of Cryptography - Volume 1 & 2* 2001 and 2004 Cambridge University Press. voir aussi en ligne : `http://www.wisdom.weizmann.ac.il/~oded/foc.html`

Shafi Goldwasser and Mihir Bellare *Lecture Notes on Cryptography* 2008
`http://cseweb.ucsd.edu/~mihir/papers/gb.html`

David Pointcheval *Le Chiffrement Asymétrique et la Sécurité Prouvée* Habilitation à diriger des recherches 2002
`https://www.di.ens.fr/~pointche/Documents/Reports/2002_HDRThesis.pdf`

Victor Shoup *Sequences of Games : A Tool for Taming Complexity in Security Proofs* 2006
`https://www.shoup.net/papers/games.pdf`

Chapitre II

Sécurité CPA

1. Introduction à la sécurité réductionniste

Pour valider la sécurité d'un système cryptographique de façon formelle, Goldwasser et Micali ont proposé en 1984 l'idée de la **sécurité réductionniste** (ou sécurité prouvée) qui tente de lier la sécurité d'un tel système à la complexité d'un problème algorithmique jugé difficile, tel que la factorisation des nombres entiers ou le calcul de logarithmes discrets.

Pour cela on prouve que si un attaquant arrive à casser un schéma cryptographique alors il peut résoudre un problème algorithmique, P_A (calculatoire ou décisionnel). Si P_C désigne le problème de casser le schéma cryptographique, on établit ainsi une réduction algorithmique polynomiale de P_A vers P_C : $P_A \propto P_C$. Ceci assure que si un attaquant est capable de casser le schéma cryptographique en temps polynomial alors il peut aussi résoudre le problème P_A en temps polynomial. On peut être plus précis en donnant les paramètres de la réduction : le temps qu'elle prend, sa probabilité de réussite, le nombre d'appel à un oracle de déchiffrement... On parle parfois de **sécurité exacte**. Ceci permet d'obtenir une **sécurité pratique** du schéma : une sélection des paramètres à utiliser pour obtenir une sécurité donnée.

On fait ensuite l'hypothèse que P_A est un problème difficile, c'est à dire qu'il n'est pas résoluble en temps polynomial. Par contraposée (de la proposition P_C est facile implique P_A est facile) on obtient donc qu'un attaquant polynomial contre le schéma cryptographique ne peut exister. C'est ce que l'on appelle une **preuve de sécurité**.

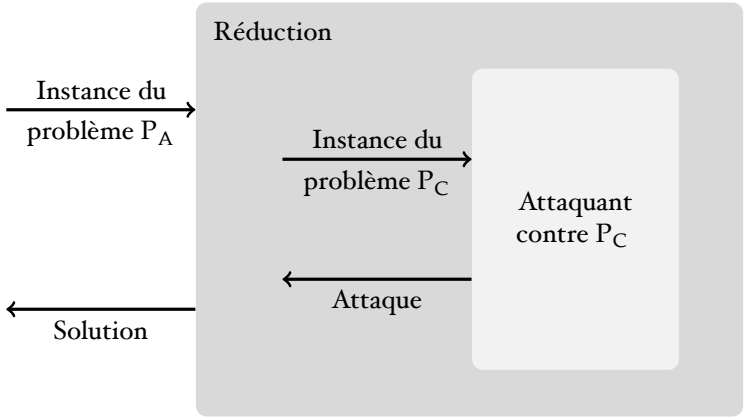


Schéma d'une preuve :

1. Hypothèse algorithmique : P_A est difficile : il ne peut être résolu par un algorithme polynomial à part avec une probabilité négligeable ;
2. Supposer l'existence d'un attaquant efficace $\mathscr{A}$ contre P_C : $\mathscr{A}$ est un algorithme polynomial probabiliste cassant le problème P_C avec bonne probabilité de succès

3. Étant donné une instance de P_A, construire un algorithme polynomial probabiliste $\mathscr{B}$, une réduction, utilisant $\mathscr{A}$ comme sous routine pour résoudre cette instance avec bonne probabilité de succès

4. Conclusion : l'existence de $\mathscr{B}$ contredit l'hypothèse algorithmique. Donc aucun attaquant efficace $\mathscr{A}$ contre P_C ne peut exister. Le schéma cryptographique est donc sûr.

Pour établir de telles preuves, il faut tout d'abord définir un modèle de sécurité : une définition formelle du schéma cryptographique et d'un adversaire contre ce schéma. On établit généralement une hiérarchie de niveaux de sécurité (reposant sur divers problèmes algorithmiques) correspondant à différents buts et moyens pour l'attaquant. Pour le chiffrement asymétrique, un attaquant a inévitablement accès à la clef publique et à l'algorithme de chiffrement, il peut obtenir le chiffrement des messages clairs de son choix. On parle d'attaques à clairs choisis ou *Chosen Plaintext Attack*, noté CPA. Ce moyen d'attaque passif fait l'objet de ce premier chapitre.

2. Fonctions à sens unique et à trappe

Les fonctions à sens unique sont des fonctions qui sont « faciles » à évaluer mais « difficile » à inverser. L'existence de telles fonctions fondamentales pour la cryptographie est une conjecture. Prouver leur existence prouverait que P ≠ NP. Le fait qu'une fonction à sens unique est « facile » à évaluer se formalise par le fait qu'il existe un algorithme polynomial d'évaluation. Pour l'inversion on requiert qu'un algorithme probabiliste polynomial $\mathscr{A}$ tentant d'inverser la fonction n'a qu'une probabilité extrêmement faible d'y arriver, la probabilité étant prise sur le choix de l'antécédent et sur les choix aléatoires de $\mathscr{A}$. Pour quantifier le « extrêmement faible », on introduit les fonctions négligeables.

Définition II – 1. Une fonction $\nu : \mathbf{N} \to \mathbf{R}^+$ est **négligeable** si pour tout $c \in \mathrm{N}$, il existe un entier k_0 tel que pour tout $k \geqslant k_0$, $\nu(k) \leqslant \frac{1}{k^c}$.

On peut de manière équivalente définir une fonction négligeable comme étant inférieure à l'inverse de tout polynôme positif à partir d'un certain rang. Si un algorithme $\mathscr{A}$ a une probabilité de succès négligeable vue comme fonction de la longueur de son entrée alors l'algorithme construit en répétant $\mathscr{A}$ un nombre polynomial de fois, aura toujours une probabilité de succès négligeable.

Définition II – 2. Une fonction $f : \{0,1\}^* \to \{0,1\}^*$ est à **sens-unique** si

- il existe un algorithme polynomial probabiliste qui sur l'entrée x calcule $f(x)$;
- pour tout algorithme polynomial probabiliste $\mathscr{A}$ il existe une fonction négligeable ν telle que pour k assez grand,

$$\mathbf{Succ}^{\mathsf{OW}}_{f,k}(\mathscr{A}) := \Pr\left[f(z) = y : x \xleftarrow{\$} \{0,1\}^k; y \leftarrow f(x); z \leftarrow \mathscr{A}(1^k, y)\right] \leqslant \nu(k)$$

La probabilité est prise sur les choix de x de k bits et les choix faits par $\mathscr{A}$. On fait dépendre $\mathscr{A}$ de 1^k pour permettre d'être polynomial en $|x|$ même si y est beaucoup plus petit que x par exemple $|y| = \log|x|$.

Notations dans la définition : Soit $k \in \mathbf{N}$, 1^k désigne k en notation unaire (1111 ... 11 avec k symboles 1). Ainsi la taille de 1^k, $|1^k| = k$. Comme en général on définit la complexité d'un algorithme en fonction de la taille de son entrée, lui donner 1^k en entrée permet d'exprimer sa complexité en fonction de k.

$x \xleftarrow{\$} \mathrm{S}$ signifie que x est tiré avec probabilité uniforme dans S.

Formellement, dans la définition, $x \xleftarrow{\$} \{0,1\}^k$ signifie que x (ou plus rigoureusement x_k) est une (famille de) variable aléatoire discrète suivant la loi uniforme sur $\{0,1\}^k$. Les autres variables y, z sont aussi à considérer comme des (familles de) variables aléatoires discrètes.

L'algorithme polynomial probabiliste $\mathscr{A}$ (l'adversaire), peut être formellement modélisé par une machine de Turing probabiliste. La sortie de $\mathscr{A}$ dépend de son entrée x mais également des choix aléatoires qu'il fait. On peut voir $\mathscr{A}$ comme une fonction déterministe (donc une fonction au sens mathématique)

prenant en entrée deux variables aléatoires : x et une chaîne de bits r de longueur suffisamment longue, uniformément distribuée et indépendante de toute autre variable (en particulier de x), rassemblant les choix aléatoires fait par $\mathscr{A}$. La sortie de $\mathscr{A}$ est aussi une variable aléatoire.

Cette définition permet de considérer une sécurité en moyenne.

Exemple : Considérer f la fonction identité et calculer le succès de $\mathscr{A}$ définit ainsi : $\mathscr{A}(1^k, y)$ tire un bit b, si $b = 0$, $\mathscr{A}$ retourne y. Si $b = 1$, $\mathscr{A}$ retourne la chaîne de bit nulle de longueur k : 0 ... 0

On peut aussi formuler le succès sous forme d'expérience : on définit l'expérience $\mathbf{Exp}^{\mathsf{OW}}_{f,k}(\mathscr{A})$:

1. Choisir l'entrée $x \xleftarrow{\$} \{0,1\}^k$, calculer $y := f(x)$
2. $\mathscr{A}$ prend 1^k et y en entrée et renvoie z
3. La sortie de l'expérience est 1 si $f(z) = y$ et 0 sinon

On a alors $\mathbf{Succ}^{\mathsf{OW}}_{f,k}(\mathscr{A}) = \Pr\left[\mathbf{Exp}^{\mathsf{OW}}_{f,k}(\mathscr{A}) = 1\right]$.

Remarques : Dans cette définition, la sécurité est asymptotique pour des tailles d'éléments x assez grand. Pour garantir une sécurité plus pratique on considère des familles de fonctions à sens-unique paramétrées par un paramètre de sécurité. Un exemple est la fonction d'exponentiation $x \to g^x$.

Les fonctions à sens-unique suffisent pour montrer l'existence de plusieurs primitives cryptographiques telles que les générateurs pseudo aléatoires, les *message authentication code*, les schémas de signatures... Pour le chiffrement asymétrique, on a besoin de plus : les fonctions à trappe.

Définition II – 3. Une **fonction à trappe** est une fonction à sens-unique $f : \{0,1\}^* \to \{0,1\}^*$ telle qu'il existe un polynôme p et un algorithme probabiliste polynomial $\mathscr{I}$ tel que pour tout entier k il existe $t_k \in \{0,1\}^*$ tel que $|t_k| \leqslant p(k)$ et pour tout $x \in \{0,1\}^k$, $\mathscr{I}(t_k, f(x)) = y$ tel que $f(y) = f(x)$.

De même on peut définir des familles de fonctions à trappe. Un exemple est la famille de fonctions RSA.

RSA (78) : Soit un nombre RSA n de k bits facteur de deux nombres premiers aléatoires distincts p et q de $k/2$ bits et e un nombre aléatoire positif premier avec $\varphi(n)$ et d l'inverse de e modulo $\varphi(n)$. Pour l'indice (n,e) on associe la trappe d et la fonction RSA $_{(n,e)} : (\mathbf{Z}/n\mathbf{Z})^\times \to (\mathbf{Z}/n\mathbf{Z})^\times : x \mapsto x^e \mod n$. C'est en fait une famille de permutations à trappe. Cette fonction est bien évaluable en temps polynomial et inversible en temps polynomial connaissant la trappe, elle est à sens-unique sous l'hypothèse RSA :

Définition II – 4. Soit un algorithme GenRSA qui prend en entrée 1^k et ressort les paramètres n, e, d de RSA. Soit $\mathscr{A}$ un attaquant, on définit l'expérience $\mathbf{Exp}^{\mathsf{OW}}_{\mathsf{GenRSA},k}(\mathscr{A})$:

1. Lancer GenRSA avec entrée 1^k pour obtenir n, e, d
2. Choisir $y \xleftarrow{\$} (\mathbf{Z}/n\mathbf{Z})^\times$ (équivalent de choisir x aléatoire)
3. $\mathscr{A}$ prend n, e, y en entrée et renvoie $z \in (\mathbf{Z}/n\mathbf{Z})^\times$
4. La sortie de l'expérience est 1 si $z^e = y$ et 0 sinon

On dit que le problème RSA est dur relativement à GenRSA si pour tout algorithme probabiliste polynomial $\mathscr{A}$ il existe une fonction négligeable ν telle que pour k assez grand,

$$\Pr\left[\mathbf{Exp}^{\mathsf{OW}}_{\mathsf{GenRSA},k}(\mathscr{A}) = 1\right] \leqslant \nu(k).$$

L'hypothèse RSA est qu'il existe un générateur GenRSA tel que le problème RSA soit dur.

3. Schéma de chiffrement asymétrique

Définition II – 5 (Schéma de chiffrement asymétrique)**.** Soit $k \in \mathbf{N}$ un paramètre de sécurité. Un schéma de chiffrement asymétrique Π est la donnée d'un triplet de trois algorithmes (KeyGen, Encrypt, Decrypt) satisfaisant les trois propriétés suivantes :

1. L'algorithme KeyGen est appelé algorithme de génération de clef. C'est un algorithme probabiliste polynomial qui prend en entrée 1^k et qui retourne un couple (pk, sk) constitué d'une clef publique, pk, et d'une clef secrète, sk. On notera $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$

2. L'algorithme Encrypt est appelé algorithme de chiffrement. C'est un algorithme polynomial probabiliste qui prend en entrée une clef publique pk et un message clair, m, élément de $\mathscr{M}$ (l'espace des messages clairs). L'algorithme Encrypt retourne un chiffré, c, élément de $\mathscr{C}$ (l'espace des chiffrés). On notera $c \leftarrow \mathsf{Encrypt}(pk, m)$

3. L'algorithme Decrypt est appelé algorithme de déchiffrement. C'est un algorithme polynomial déterministe qui prend en entrée une clef secrète sk et un chiffré c élément de $\mathscr{C}$ et qui retourne la valeur $\mathscr{D}(sk, c)$. Cette valeur sera soit un message clair élément de $\mathscr{M}$, soit une erreur notée $\perp$ qui indique que le chiffré n'est pas valide.

Le schéma doit être correct, c'est à dire que pour tout paramètre de sécurité k, et pour tout message $m \in \mathscr{M}$, si $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$, alors

$$\mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, m)) = m$$

avec probabilité 1. La probabilité étant sur tous les choix aléatoires que font les algorithmes KeyGen et Encrypt.

Exemples :

1. Le schéma de chiffrement trivial où $\mathsf{Encrypt}(pk, m) = m$ et $\mathsf{Decrypt}(sk, c) = c$: il faut définir des notions de sécurité !

2. Une famille de permutations à trappe permet de définir un système de chiffrement. KeyGen sélectionne une fonction trappe f comme clef publique, sa trappe t_k et l'algorithme $\mathscr{I}$ comme clef privée. À tout message m, $\mathsf{Encrypt}(f, m) = f(m)$. Étant donné c et t_k, $\mathsf{Decrypt}(t_k, c) := \mathscr{I}(t_k, c) = f^{-1}(c)$. Avec la permutation à trappe RSA, cela donne le système *textbook* RSA.

La sécurité de base pour un système de chiffrement est le **bris total**. Celui ci est atteint si l'attaquant est capable de retrouver la clef privée au moyen de la clef publique. On note TB – CPA une telle attaque. Un système sera donc sûr contre le bris total si un attaquant probabiliste polynomial n'a qu'une probabilité négligeable d'y arriver. Pour RSA cela revient à retrouver la trappe d, ce qui est équivalent à factoriser n. RSA est donc TB – CPA sûr sous l'hypothèse que la factorisation est difficile.

Une sécurité plus forte est la notion de sens-unique :

Définition II – 6 (OW – CPA)**.** Soit $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ un schéma de chiffrement asymétrique. Soit $\mathscr{A}$ un algorithme attaquant Π. On définit sa probabilité de succès dans l'inversion ponctuelle du schéma par

$$\mathbf{Succ}_{\Pi}^{\mathsf{OW}}(\mathscr{A}) := \Pr\Big[\mathscr{A}(pk, c) = m : (pk, sk) \leftarrow \mathsf{KeyGen}(1^k); m \xleftarrow{\$} \mathscr{M}; c \leftarrow \mathsf{Encrypt}(pk, m)\Big].$$

On dira que le schéma Π est sûr au sens OW – CPA si ce succès est négligeable pour tout algorithme polynomial probabiliste $\mathscr{A}$.

Cette sécurité est garantie si on utilise une famille de fonction à trappe comme dans le second exemple. Elle n'est cependant pas suffisante. Le fait que la fonction de chiffrement f utilisée soit à sens-unique ne garantie pas que $f(x)$ cache toutes les informations sur x et l'attaquant peut se contenter d'une information partielle sur le message clair. Ce problème est flagrant si la fonction de chiffrement est déterministe : supposons que $\mathscr{M} = \{0,1\}$ alors un adversaire peut calculer les chiffrés de 0 et de 1 et les reconnaître dans un message chiffré. Ainsi le *textbook* RSA vu plus haut n'atteint pas un niveau de sécurité suffisant pour être utilisé tel quel en pratique.

4. Sécurité sémantique

En cryptographie asymétrique, la sécurité inconditionnelle ou parfaite au sens de Shannon n'est pas possible. La notion de **sécurité sémantique** (ou sécurité polynomiale) est qu'un attaquant ne puisse extraire aucune information en temps polynomial sur un message clair à partir de l'un des chiffrés, en dehors de celles qu'il aurait pu obtenir sans ce chiffré (notion introduite par Goldwasser et Micali en 1984).

Définition II – 7 (Sécurité sémantique CPA)**.** Soit $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ un schéma de chiffrement asymétrique. Soit $\mathscr{A}$ un algorithme probabiliste polynomial attaquant Π et k un paramètre de sécurité, et soit $\mathscr{E}nv$ un algorithme polynomial probabiliste représentant le contexte. On définit l'expérience de sécurité sémantique CPA, $\mathbf{Exp}^{\mathsf{sem-sec-CPA}}_{\Pi,k}(\mathscr{A})$:

1. On lance l'algorithme $\mathsf{KeyGen}(1^k)$ pour obtenir les clefs (pk, sk)
2. On donne la clef pk à $\mathscr{E}nv$ qui retourne un message m, une relation binaire R et une information publique partielle (donnée par le contexte) α
3. On calcule c un chiffré de m : $c \leftarrow \mathsf{Encrypt}(pk, m)$ le ***challenge*** et on donne (c, α, pk) à $\mathscr{A}$
4. $\mathscr{A}$ retourne une information z
5. La sortie de l'expérience est 1 si z est en relation avec m, c'est à dire si $\mathrm{R}(m, z)$ et 0 sinon.

Soit $\mathscr{S}$ un algorithme probabiliste polynomial, jouant le rôle de simulateur On définit maintenant l'expérience de sécurité sémantique CPA simulée, $\mathbf{Exp}^{\mathsf{sem-sec-simul-CPA}}_{\Pi,k}(\mathscr{S})$:

1. On lance l'algorithme $\mathsf{KeyGen}(1^k)$ pour obtenir les clefs (pk, sk)
2. On donne la clef pk à $\mathscr{E}nv$ qui retourne un message m, une relation binaire R et une information publique partielle α
3. On donne (α, pk) à $\mathscr{S}$
4. $\mathscr{S}$ retourne une information z
5. La sortie de l'expérience est 1 si z est en relation avec m, c'est à dire si $\mathrm{R}(m, z)$ et 0 sinon.

On suppose que les points 1 et 2 des deux expériences sont communs. Le schéma Π est sûr au sens $\mathsf{sem-sec-CPA}$ si pour tous les algorithmes polynomiaux probabilistes $\mathscr{A}$ et $\mathscr{E}nv$, il existe un algorithme probabiliste polynomial $\mathscr{S}$ et une fonction négligeable ν telle que pour k assez grand,

$$\left|\Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) - \Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-simul-CPA}}_{\Pi,k}(\mathscr{S}) = 1\right)\right| \leqslant \nu(k)$$

Dans la définition l'algorithme $\mathscr{E}nv$ permet de préciser le contexte dans lequel l'émetteur, le récepteur et l'adversaire évoluent : un événement force l'émission d'un message chiffré. Le clair correspondant est généré par $\mathscr{E}nv$ ainsi qu'une information publique α, représentant le contexte, que connaîtra l'adversaire. L'algorithme $\mathscr{E}nv$ produit également une relation binaire R : l'adversaire va essayer de deviner à partir du

chiffré une information $z \in \mathscr{Z}$ sur le message clair et sera gagnant si z est bien en relation avec m. C'est à dire que $R \subset \mathscr{M} \times \mathscr{Z}$ et $R(m, z)$.

Un exemple : Oscar sait qu'Alice et Bob sont dans la même ville aujourd'hui, qu'ils doivent se rencontrer et qu'Alice va envoyer un message chiffré à Bob pour préciser où se rencontrer : le nom de la rue m. Toute cette information publique est représentée par α. Une information z sur m que peut essayer de trouver Oscar est la première lettre du nom de la rue. On dira que m et z sont en relation et qu'Oscar a gagné si z représente la première lettre de m.

Comme dit précédemment, on cherche à quantifier l'information obtenue sur m à partir de c comparé à celle que l'on aurait pu obtenir sans c, à partir du seul contexte. L'expérience simulée mesure la probabilité d'obtenir l'information avec le seul contexte (les 3/4 des rues de la ville commencent par la lettre S...). Ainsi, l'adversaire réussit vraiment son attaque s'il obtient une réelle information, c'est à dire si sa probabilité de succès n'est pas trop proche de celle de la simulation et qu'elle est donc exploitable en pratique.

Dans la pratique, on utilise une notion moins naturelle mais bien plus simple à manipuler, la notion d'**indistinguabilité** : si m_0 et m_1 sont deux messages clairs et si c est un chiffré de l'un de ces deux messages, alors un attaquant ne peut pas décider si c a été retourné par $\mathsf{Encrypt}(pk, m_0)$ ou par $\mathsf{Encrypt}(pk, m_1)$, toujours en temps polynomial. Cette notion, introduite aussi par Goldwasser et Micali est équivalente à la notion de sécurité sémantique (voir plus loin).

Définition II – 8 (IND – CPA)**.** Soit $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ un schéma de chiffrement asymétrique. Soit $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ un algorithme attaquant Π et k un paramètre de sécurité. On définit l'expérience d'indistinguabilité CPA, $\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A})$:

1. On lance l'algorithme $\mathsf{KeyGen}(1^k)$ pour obtenir les clefs (pk, sk)
2. $\mathscr{A}_1$ reçoit pk et retourne (m_0, m_1, s) avec deux messages de $\mathscr{M}$ de même longueur et un état d'information s
3. On choisit un bit aléatoire $b^\star \xleftarrow{\$} \{0,1\}$
4. On calcule $c^\star$ un chiffré de $m_{b^\star}$: $c^\star \leftarrow \mathsf{Encrypt}(pk, m_{b^\star})$: c'est le *challenge* et on donne $(s, c^\star)$ à $\mathscr{A}_2$
5. $\mathscr{A}_2$ sort un bit b
6. La sortie de l'expérience est 1 si $b = b^\star$ et 0 sinon.

L'avantage de l'attaquant $\mathscr{A}$ pour résoudre l'indistinguabilité du schéma Π est défini par

$$\mathbf{Adv}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A}) = \left|\Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) - \frac{1}{2}\right|.$$

Le schéma Π est sûr au sens IND – CPA si pour tout algorithme polynomial probabiliste $\mathscr{A}$ il existe une fonction négligeable ν telle que pour k assez grand,

$$\mathbf{Adv}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A}) \leqslant \nu(k)$$

Remarques

1. L'état d'infomation s permet de « transmettre » l'état de $\mathscr{A}_1$ à $\mathscr{A}_2$. En particulier, on peut supposer que s contient toujours les messages m_0 et m_1.
2. Dans la définition précédente, le terme avantage signifie l'avantage de l'algorithme $\mathscr{A}$ par rapport à un lancer de pièce : si $\mathscr{A}_2$ tire au hasard sa réponse, on aura $\mathbf{Adv}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A}) = 0$.
3. Soit Π un schéma de chiffrement, si l'algorithme Encrypt est déterministe alors Π n'est pas sûr au sens IND – CPA.

4. On définit parfois l'avantage de la manière suivante :

$$\mathbf{Adv}^{\mathsf{IND-CPA}'}_{\Pi,k}(\mathscr{A}) = \left|2\Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) - 1\right|.$$

soit deux fois l'avantage précédent (cela ne change pas grand chose pour des considérations asymptotiques). En utilisant les formules suivantes (on allège les notations)

$$\begin{aligned}\Pr\left[b = b^\star\right] &= \Pr\left[b = b^\star | b^\star = 1\right] \times \Pr\left[b^\star = 1\right] + \Pr\left[b = b^\star | b^\star = 0\right] \times \Pr\left[b^\star = 0\right] \\ &= \frac{\Pr\left[b = 1 | b^\star = 1\right]}{2} + \frac{\Pr\left[b = 0 | b^\star = 0\right]}{2},\end{aligned}$$

et

$$\Pr\left[b = 1 | b^\star = 0\right] + \Pr\left[b = 0 | b^\star = 0\right] = 1,$$

on voit facilement qu'avec cette deuxième notation de l'avantage on a

$$\begin{aligned}\mathbf{Adv}^{\mathsf{IND-CPA}'}_{\Pi,k}(\mathscr{A}) &= \left|\Pr\left[b = 1 | b^\star = 1\right] + \Pr\left[b = 0 | b^\star = 0\right] - 1\right| \\ &= \left|\Pr\left[b = 1 | b^\star = 1\right] - \Pr\left[b = 1 | b^\star = 0\right]\right| \\ &= \left|\Pr\left[b = 0 | b^\star = 1\right] - \Pr\left[b = 0 | b^\star = 0\right]\right|\end{aligned}$$

C'est à dire l'avantage de $\mathscr{A}$ pour distinguer entre les deux distributions : les chiffrés tel que $b^\star = 1$ et ceux tel que $b^\star = 0$.

Théorème II – 9 (Goldwasser - Micali (84))**.** Un schéma de chiffrement asymétrique est sémantiquement sûr CPA si et seulement si il est IND – CPA sûr.

Démonstration. On suit une preuve de Dodis.

On commence par montrer que IND – CPA ⇒ sémantiquement sûr CPA.

Soit un schéma Π sûr au sens IND – CPA. On suppose qu'il n'est pas sémantique sûr, c'est à dire qu'il existe un adversaire $\mathscr{A}$ contre cette notion. On va construire un adversaire $\mathscr{B}$ contre l'IND – CPA. Si cet adversaire est efficace, on obtiendra une contradiction donc Π est sémantiquement sûr.

Il existe donc deux algorithmes probabilistes polynomiaux $\mathscr{E}nv$ et $\mathscr{A}$ tel que pour tout algorithme probabiliste polynomial simulateur $\mathscr{S}$,

$$\left|\Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) - \Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-simul-CPA}}_{\Pi,k}(\mathscr{S}) = 1\right)\right| \geqslant \epsilon$$

où ϵ est non négligeable. En particulier on peut prendre $\mathscr{S}$ comme suit :

1. $\mathscr{S}$ reçoit (α, pk)
2. Soit c' un chiffré de m' : $c' \leftarrow \mathsf{Encrypt}(pk, m')$ où m' est un message fixe de $\mathscr{M}$ de même longueur que le message m choisit par $\mathscr{E}nv$ (par exemple le message avec tous les bits à 0)
3. $\mathscr{S}$ renvoie $z' \leftarrow \mathscr{A}(c', \alpha, pk)$

Pour résumer on a

1. $(pk, sk) \leftarrow \mathsf{KeyGen}(1^k)$
2. $(m, \mathrm{R}, \alpha) \leftarrow \mathscr{E}nv(pk)$
3. $c \leftarrow \mathsf{Encrypt}(pk, m)$ et $c' \leftarrow \mathsf{Encrypt}(pk, m')$

4. $z \leftarrow \mathscr{A}(c, \alpha, pk)$ et $z' \leftarrow \mathscr{A}(c', \alpha, pk)$

et par hypothèse,

$$|\Pr(\mathrm{R}(m, z)) - \Pr(\mathrm{R}(m, z'))| \geqslant \epsilon.$$

L'algorithme $\mathscr{A}$ va nous permettre de distinguer les chiffrés de m et ceux de m'. On construit maintenant un attaquant probabiliste polynomial $\mathscr{B} = (\mathscr{B}_1, \mathscr{B}_2)$ contre IND – CPA à partir de $\mathscr{A}$.

On définit $\mathscr{B}_1$:

1. $\mathscr{B}_1$ reçoit pk
2. Il obtient (m, R, α) de $\mathscr{E}nv(pk)$
3. Il ressort (m, m', s) avec $s = (\mathrm{R}, \alpha, pk)$

On définit maintenant $\mathscr{B}_2$:

1. $\mathscr{B}_2$ reçoit $(s, c^\star)$ avec $s = (\mathrm{R}, \alpha, pk)$
2. On obtient $z^\star$ de $\mathscr{A}(c^\star, \alpha, pk)$
3. Si $\mathrm{R}(m, z^\star)$, $\mathscr{B}_2$ sort $b = 0$ (le message est m), sinon $\mathscr{B}_2$ sort $b = 1$ (le message est m').

On note $b^\star$ le bit tiré lors de l'expérience d'indistinguabilité. On a $b^\star = 0$ si $c^\star$ est un chiffré de m et $b^\star = 1$ si c'est un chiffré de m'.

Si $b^\star = 0$, le message est m, $\mathscr{A}$ ressort $z^\star = z$ et $\mathscr{B}_2$ ressort aussi $b = 0$ si $\mathrm{R}(m, z)$ ce qui arrive avec $\Pr(\mathrm{R}(m, z)) = \Pr(b = b^\star | b^\star = 0)$.

Si $b^\star = 1$, le message est m', $\mathscr{A}$ ressort $z^\star = z'$ et $\mathscr{B}_2$ ressort aussi $b = 1$ si $\neg\mathrm{R}(m, z')$ ce qui arrive avec $1 - \Pr(\mathrm{R}(m, z')) = \Pr(b = b^\star | b^\star = 1)$. Au final, la probabilité que l'expérience d'indistinguabilité soit un succès est

$$\Pr(b = b^\star) = \frac{1}{2}\Pr(\mathrm{R}(m, z)) + \frac{1}{2}(1 - \Pr(\mathrm{R}(m, z'))),$$

ce qui donne $\frac{1}{2} + \frac{\Pr(\mathrm{R}(m,z)) - \Pr(\mathrm{R}(m,z'))}{2}$. Donc l'avantage

$$\mathbf{Adv}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{B}) = \left| \frac{\Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) - \Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-simul-CPA}}_{\Pi,k}(\mathscr{S}) = 1\right)}{2} \right|$$

ce qui est non négligeable.

Réciproque : on montre que sémantiquement sûr CPA $\Rightarrow$ IND – CPA.

On suppose donc que Π est sémantiquement sûr CPA et qu'il n'est pas IND – CPA. Soit $\mathscr{B} = (\mathscr{B}_1, \mathscr{B}_2)$ un attaquant IND – CPA ayant un avantage non négligeable. On peut supposer que les messages m_0, m_1 choisis par $\mathscr{B}_1$ sont distincts. Cet attaquant peut être vu comme un attaquant contre la sécurité sémantique qui étant donné le contexte suivant « Alice va envoyer un chiffré de m_0 ou de m_1 », obtient, à partir du chiffré, l'information « être un chiffré de m_0 ou de m_1 ». C'est à dire satisfaire une relation d'égalité. Plus précisément, on construit $\mathscr{E}nv$ comme suit :

1. $\mathscr{E}nv$ reçoit pk
2. On donne pk à $\mathscr{B}_1$ qui retourne (m_0, m_1, s) : $(m_0, m_1, s) \leftarrow \mathscr{B}_1(pk)$
3. On pose $\alpha = (m_0, m_1, s)$ et R la relation $\mathrm{R}(m, z) \Leftrightarrow m = z$.
4. On tire $b^\star \xleftarrow{\$} \{0, 1\}$, et on renvoie $(m_{b^\star}, \mathrm{R}, \alpha)$

On définit ensuite l'attaquant $\mathscr{A}$ contre la sécurité sémantique :

1. $\mathscr{A}$ reçoit $(c^\star, \alpha, pk)$ avec $\alpha = (m_0, m_1, s)$

2. On donne $(s, c^\star)$ à $\mathscr{B}_2$ qui retourne un bit b
3. On retourne $z := m_b$

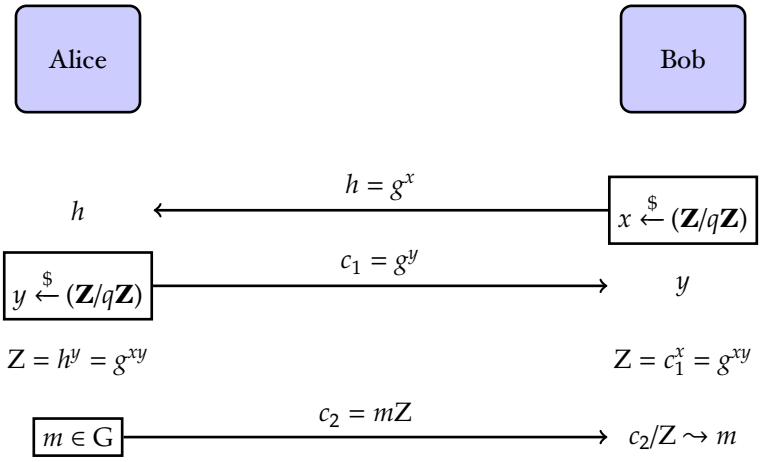
Il est clair que la probabilité $\Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) = \Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{B}) = 1\right)$. Par contre tout algorithme $\mathscr{S}$ qui n'aurait accès qu'à (α, pk) n'aurait aucune information sur le bit $b^\star$ tiré par $\mathscr{E}nv$ ($b^\star$ est déterminé après α). Cet algorithme $\mathscr{S}$ n'a donc qu'une chance sur deux de retourner le bon message. Ainsi, $\Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-simul-CPA}}_{\Pi,k}(\mathscr{S}) = 1\right) = \frac{1}{2}$. Au final, on a

$$\left|\Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) - \Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-simul-CPA}}_{\Pi,k}(\mathscr{S}) = 1\right)\right| =$$

$$\left|\Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{B}) = 1\right) - \frac{1}{2}\right| = \mathbf{Adv}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{B}),$$

qui est non négligeable. □

Exemple Elgamal (85) : Soit GenDH un algorithme polynomial qui prend en entrée 1^k et retourne la description d'un groupe cyclique $\mathbf{G}$ son ordre q (premier avec $|q| = k$ et un générateur g. On suppose que l'on peut calculer dans G en temps polynomial. L'algorithme KeyGen appelle GenDH puis choisit x aléatoire dans $\mathbf{Z}/q\mathbf{Z}$ et calcule $h = g^x$. KeyGen retourne $pk = (\mathbf{G}, q, g, h)$ et $sk = (\mathrm{G}, q, g, x)$. On pose $\mathscr{M} = \mathrm{G}$ et $\mathscr{C} = \mathrm{G}\times\mathrm{G}$. L'algorithme Encrypt sur l'entrée (pk, m) choisit y aléatoirement dans $\mathbf{Z}/q\mathbf{Z}$ et retourne $c = (g^y, mh^y)$. L'algorithme Decrypt sur l'entrée $(sk, (c_1, c_2))$ retourne c_2/c_1^x. On peut voir la deuxième coordonnée d'un chiffré Elgamal $m \mapsto mh^y$ comme un *one-time pad* dans $\mathbf{G}$. Si y est choisi de manière uniforme mh^y n'apporte aucune information sur m, on a donc un chiffrement parfait. Mais comme g^y et h sont aussi donnés, ce n'est pas le cas. En fait, h^y peut être vu comme une clef secrète partagée par un échange de clef Diffie-Hellman ne servant qu'une fois :



où Z est une clef secrète établie par l'échange de clef Diffie-Hellman, utilisée ensuite pour chiffrer m par un *one time pad*.

On définit les problèmes CDH et DDH sur lesquels reposent la sécurité d'Elgamal.

Définition II – 10. Soit $\mathscr{A}$ un attaquant, on définit l'expérience $\mathbf{Exp}^{\mathsf{CDH}}_{\mathsf{GenDH},k}(\mathscr{A})$:

1. Lancer GenDH avec entrée 1^k pour obtenir G, q, g
2. Choisir $x, y \xleftarrow{\$} (\mathbf{Z}/q\mathbf{Z})$, et calculer $\mathrm{X} = g^x$ et $\mathrm{Y} = g^y$
3. $\mathscr{A}$ prend $\mathrm{G}, q, g, (\mathrm{X}, \mathrm{Y})$ en entrée et renvoie $\mathrm{Z} \in \mathrm{G}$

4. La sortie de l'expérience est 1 si $Z = g^{xy}$ et 0 sinon

On dit que le problème CDH est dur relativement à GenDH si pour tout algorithme probabiliste polynomial $\mathscr{A}$ il existe une fonction négligeable ν telle que pour k assez grand, le succès

$$\Pr\left[\mathbf{Exp}^{\mathsf{CDH}}_{\mathsf{GenDH},k}(\mathscr{A}) = 1\right] \leqslant \nu(k).$$

L'hypothèse CDH (*computational Diffie–Hellman*) est qu'il existe un générateur GenDH tel que le problème CDH soit dur.

Définition II – 11. Soit $\mathscr{D}$ un attaquant, on définit l'expérience $\mathbf{Exp}^{\mathsf{DDH}}_{\mathsf{GenDH},k}(\mathscr{D})$:

1. Lancer GenDH avec entrée 1^k pour obtenir G, q, g
2. Choisir $x, y \xleftarrow{\$} (\mathbf{Z}/q\mathbf{Z})$, et calculer $X := g^x$ et $Y := g^y$ et prendre $b^\star \xleftarrow{\$} \{0,1\}$
3. Si $b^\star = 0$ alors $Z \xleftarrow{\$} G$, sinon $Z := g^{xy}$
4. $\mathscr{D}$ prend $G, q, g, (X, Y, Z)$ en entrée et renvoie un bit b
5. La sortie de l'expérience est 1 si $b = b^\star$ et 0 sinon

On dit que le problème DDH est dur relativement à GenDH si pour tout algorithme probabiliste polynomial $\mathscr{D}$ il existe une fonction négligeable ν telle que pour k assez grand,

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathsf{GenDH},k}(\mathscr{D}) = \left|\Pr\left(\mathbf{Exp}^{\mathsf{DDH}}_{\mathsf{GenDH},k}(\mathscr{D}) = 1\right) - \frac{1}{2}\right| \leqslant \nu(k).$$

L'hypothèse DDH (*Decision Diffie–Hellman*) est qu'il existe un générateur GenDH tel que le problème DDH soit dur.

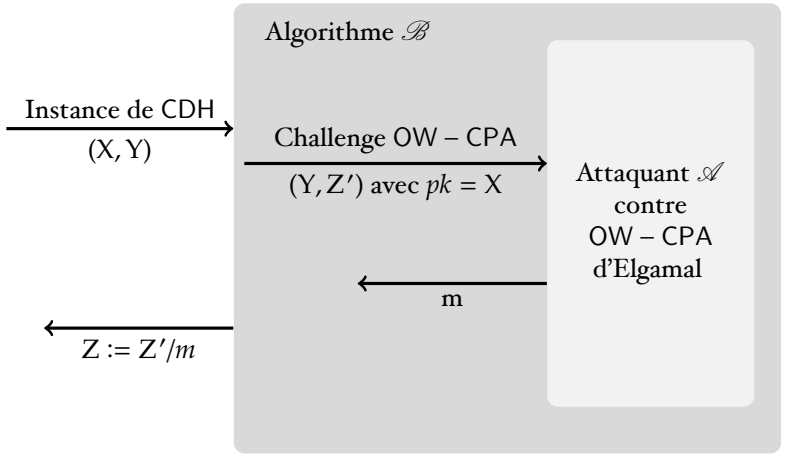
On conjecture que cette hypothèse DDH est vraie dans les sous groupes de $(\mathbf{Z}/p\mathbf{Z})^\times$ d'ordre q avec q un grand diviseur premier de $p-1$ avec p premier (par exemple en utilisant des nombres premiers de Sophie Germain : $p = 2q+1$, on travaille dans ce cas avec les carrés de $(\mathbf{Z}/p\mathbf{Z})^\times$. On utilise aussi certaines courbes elliptiques (en particulier, il ne faut pas qu'un couplage puisse être calculé : étant donné P, aP, bP, cP, on a $e(P, cP) = e(aP, bP)$ si et seulement si $c = ab$ modulo l'ordre de P).

Théorème II – 12. Elgamal est OW – CPA sous l'hypothèse CDH et IND – CPA sous l'hypothèse DDH.

Démonstration. On commence par la sécurité OW–CPA. On suppose donc que Elgamal n'est pas OW–CPA et on construit un algorithme résolvant le problème CDH.

On se donne G, q, g retournés par GenDH. On note $X = g^x$ et $Y = g^y$ le début du triplet Diffie-Hellman à reconstituer. On prend $pk = (\mathbf{G}, q, g, X)$ comme clef publique pour un chiffrement Elgamal : X étant un élément de G tiré uniformément, on obtient bien la même distribution que la clef publique. On construit ensuite le chiffré (Y, Z') où Z' est pris aléatoirement dans G. C'est bien un chiffré valide d'un message aléatoire : c'est évident pour la première coordonnée, et pour la deuxième coordonnée, prendre un message $m \xleftarrow{\$} G$ et calculer $Z' = mX^y$ revient à prendre directement $Z' \xleftarrow{\$} G$.

Comme Elgamal est supposé non OW – CPA, il existe un attaquant $\mathscr{A}$ retrouvant le message m dont (Y, Z') est un chiffré avec un succès non négligeable. Mais on aura alors $m = Z'/Y^x$, donc $Z := Z'/m = Y^x = g^{xy}$. On a donc construit un algorithme $\mathscr{B}$ résolvant CDH avec le même succés non-négligeable. On a donc une contradiction. On résume la construction de $\mathscr{B}$ par la figure suivante (bien noter que la vue de $\mathscr{A}$ est bien un challenge OW – CPA pour Elgamal, avec la bonne distribution).



On montre maintenant la sécurité IND – CPA. On procède toujours par l'absurde. Soit $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ un adversaire IND – CPA avec un avantage non négligeable. Soit (X, Y, Z) un challenge DDH. On construit un algorithme $\mathscr{D}$ comme suit :

1. Entrée : G, q, g, (X, Y, Z)
2. On pose $pk = (\mathbf{G}, q, g, \mathrm{X})$ comme clef publique
3. On obtient m_0, m_1, s par $\mathscr{A}_1(pk)$
4. On tire un bit $b^\star$
5. On construit le chiffré $c^\star = (\mathrm{Y}, m_{b^\star}\mathrm{Z})$
6. On donne $c^\star$ avec s à $\mathscr{A}_2$ qui retourne un bit b
7. On retourne 1 si $b = b^\star$ et 0 sinon

Dans le cas où (X, Y, Z) n'est pas un triplet Diffie-Hellman, comme Z est uniformément distribuée dans G indépendante des autres variables aléatoires, la seconde composante du chiffré $c^\star$, $m_{b^\star}$Z est aussi uniformément distribuée dans G et est indépendante de $m_{b^\star}$ (*one-time pad*). Le chiffré $c^\star$, en particulier, et toutes les entrées de l'adversaire, en général, sont donc indépendantes du bit $b^\star$. Même un adversaire avec une puissance de calcul infini ne peut donc deviner le bit $b^\star$ qu'avec un avantage nul :

$$\Pr[b = b^\star] = \Pr[b = 0 | b^\star = 0]\Pr[b^\star = 0] + \Pr[b = 1 | b^\star = 1]\Pr[b^\star = 1] = \frac{1}{2}(\Pr[b = 0] + \Pr[b = 1]) = \frac{1}{2}.$$

Ainsi $\mathscr{D}$ retourne 0 ou 1 avec probabilité 1/2.

Maintenant si (X, Y, Z) est un triplet valide, $c^\star$ est un chiffré valide de $m^\star$. La vue de $\mathscr{A}$ est donc la même que dans l'expérience d'indistinguabilité. Le distingueur $\mathscr{D}$ retourne donc 1 avec probabilité

$$\Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\mathsf{Elgamal},k}(\mathscr{A}) = 1\right).$$

Au final, si b' désigne le bit choisi dans l'expérience DDH, l'expérience retourne 1 sachant que $b' = 0$ avec probabilité 1/2, et retourne 1 sachant que $b' = 1$ avec probabilité $\Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\mathsf{Elgamal},k}(\mathscr{A}) = 1\right)$.

On a donc

$$\Pr\left(\mathbf{Exp}^{\mathsf{DDH}}_{\mathsf{GenDH},k}(\mathscr{D}) = 1\right) = \frac{1}{2} \times \frac{1}{2} + \Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\mathsf{Elgamal},k}(\mathscr{A}) = 1\right) \times \frac{1}{2}.$$

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathsf{GenDH},k}(\mathscr{D}) = \frac{1}{2}\left|\Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\mathsf{Elgamal},k}(\mathscr{A}) = 1\right) - \frac{1}{2}\right|$$

donc

$$\mathbf{Adv}^{\mathsf{DDH}}_{\mathsf{GenDH},k}(\mathscr{D}) = \frac{1}{2}\mathbf{Adv}^{\mathsf{IND-CPA}}_{\mathsf{Elgamal},k}(\mathscr{A})$$

qui est donc non négligeable.

□

5. Les preuves par jeux

Les preuves de sécurité que nous avons vu jusqu'à présent sont relativement simples. Cependant lorsque l'on considère des protocoles cryptographiques plus avancés que le chiffrement ou des notions de sécurité plus fortes, les preuves s'allongent et il devient difficile pour le lecteur de vérifier que tous les arguments ont bien été énoncé pour le calcul final d'avantage. Aussi, Victor Shoup a formalisé en 2006 le concept des preuves par jeux, permettant une présentation uniforme des preuves de sécurité et de faire apparaître clairement les arguments les uns après les autres.

L'idée est la suivante. On considère d'abord un jeu 0 correspondant à l'expérience définissant la sécurité (par exemple l'expérience IND – CPA). À ce jeu on associe un évènement S_0 dont la probabilité intervient dans la définition du modèle de sécurité (par exemple $b = b^\star$ pour l'IND – CPA). On modifie ensuite de manière minimale ce jeu pour construire des jeux $1, 2, \dots, n$. On note S_i l'évènement défini au jeu i, avec une définition proche de celle de S_0. L'idée est que $\Pr(S_n)$ soit égale à une probabilité cible (par exemple 1/2 pour l'IND – CPA) et que $|\Pr(S_i) - \Pr(S_{i+1})|$ soit négligeable pour $i = 0, \dots, n-1$. Ainsi par inégalité triangulaire, on aura $|\Pr(S_0) - \Pr(S_n)|$ négligeable et le résultat de sécurité.

Pour passer d'un jeu à un autre, on suit en général trois types de transitions :

1. Des transitions consistant à une réécriture équivalente des variables (par exemple $Z = g^{xy}$ au lieu de $Z = c_1^x$ et $c_1 = g^y$), dans ce cas là, $\Pr(S_i) = \Pr(S_{i+1})$.
2. Des transitions basées sur deux distributions D_1, D_2 indistinguables (par exemple (g^x, g^y, g^{xy}) et (g^x, g^y, g^z)) : le jeu i va utiliser des variables de D_1 et le jeu $i+1$, des variables de D_2. Pour prouver que $|\Pr(S_i) - \Pr(S_{i+1})|$ est négligeable, on va prouver qu'il existe un algorithme qui sort 1 avec probabilité $\Pr(S_i)$ (resp. $\Pr(S_{i+1})$) étant donné un élément tiré selon la distribution D_1 (resp. D_2).
3. Une transition basée sur l'occurrence d'un évènement « d'échec », F : les deux jeux procèdent similairement à moins que l'évènement F arrive, c'est à dire que $S_i \wedge \neg F$ est équivalent à $S_{i+1} \wedge \neg F$. On voit alors facilement que $|\Pr(S_i) - \Pr(S_{i+1})| \leqslant \Pr(F)$:

$$\begin{aligned} |\Pr(S_i) - \Pr(S_{i+1})| &= |\Pr(S_i \wedge F) + \Pr(S_i \wedge \neg F) - \Pr(S_{i+1} \wedge F) - \Pr(S_{i+1} \wedge \neg F)| \\ &= |\Pr(S_i \wedge F) - \Pr(S_{i+1} \wedge F)| \\ &\leqslant \Pr(F) \end{aligned}$$

où la dernière égalité vient du fait que les deux probabilités $\Pr(S_i \wedge F)$ et $\Pr(S_{i+1} \wedge F)$ sont comprises entre 0 et $\Pr(F)$.

Comme exemple, voyons comment réécrire la preuve IND – CPA d'Elgamal dans ce formalisme, on va essentiellement retrouver notre preuve précédente, mais présentée différemment.

On commence par écrire un jeu 0 qui est l'expérience IND–CPA d'Elgamal et on lui associe l'évènement $S_0 : b = b^\star$. Ainsi

$$\Pr(S_0) = \Pr\Big(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A}) = 1\Big).$$

Ensuite on écrit un jeu 1, similaire au jeu 0 mais on modifiant le calcul du chiffré challenge $c^\star$. Au lieu de calculer $c_2^\star = m_{b^\star} h^y$ où $h = g^x$ est la clef publique, comme dans le jeu 0, on calcule $c_2^\star = m_{b^\star} Z$ avec Z uniforme dans G. On montre ensuite que $Pr(S_1) = 1/2$ car dans le jeu 1, l'adversaire n'a plus d'information sur $b^\star$.

Puis on étudie la différence $|\Pr(S_0) - \Pr(S_1)|$. Au jeu 0, $h = g^x, c_1 = g^y, h^y$ est un triplet Diffie-Hellman, alors qu'au jeu 1, $h = g^x, c_1 = g^y, Z$ est un triplet aléatoire de G^3. On peut alors construire un distingueur prenant en entrée un tel triplet et dont l'exécution correspond au jeu 0 (resp. au jeu 1) si c'est

un triplet Diffie-Hellman (resp. un triplet aléatoire). C'est exactement le distingueur vu dans la preuve IND – CPA d'Elgamal. On a alors $\Pr(S_0)$ (resp. $\Pr(S_1)$) égale à la probabilité que le distingueur retourne 1 étant donné un triplet Diffie-Hellman (resp. un triplet aléatoire). On voit facilement que $|\Pr(S_0) - \Pr(S_1)| = 2\mathbf{Adv}^{\mathsf{DDH}}_{\mathsf{GenDH},k}(\mathscr{D})$, ce qui permet de conclure que

$$\mathbf{Adv}^{\mathsf{IND-CPA}}_{\mathsf{Elgamal},k}(\mathscr{A}) = |\Pr(S_0) - \Pr(S_1)|,$$

est négligeable.

6. Non malléabilité

Le chiffrement Elgamal est malléable : si (c_1, c_2) est un chiffré de la forme (g^y, mh^y) alors, $(c_1, c_2) \times (g^r, m'h^r)$ est un chiffré de mm'. On dit qu'Elgamal est **homomorphe multiplicatif**. Cette propriété peut-être recherché pour des applications comme on le verra dans la section suivante. Cela peut aussi être considéré comme une faiblesse, un attaquant peut ainsi mettre à mal l'intégrité du message clair en transformant (c_1, c_2) en un chiffrement d'un autre message. Ceci est d'autant plus nuisible dans le cas d'attaque active. Une autre notion de sécurité est donc la non malléabilité (NM) introduite par Dolev, Dwork et Naor (91).

De manière informelle, on ne veut pas qu'un attaquant étant donné un chiffré c de m puisse produire un nouveau chiffré c' d'un message m' telle que m et m' soient en relation. On peut formaliser cette notion avec un attaquant à deux étapes comme pour l'indistinguabilité. On montre alors que NM – IND $\Rightarrow$ IND – CPA. Dans le cas d'attaque actives adaptatives, on a une équivalence entre les deux notions.

7. Résumé

On a

$$\mathsf{TB-CPA} \Longleftarrow \mathsf{OW-CPA} \Longleftarrow \begin{array}{c}\mathsf{IND-CPA} \\ \Updownarrow \\ \mathsf{sem-sec-CPA}\end{array} \Longleftarrow \mathsf{NM-CPA}$$

On a des séparations pour chaque notion de sécurité de la première ligne, ainsi on peut construire des schémas de chiffrements :

- TB – CPA mais non OW – CPA : $sk = \$$, $pk = \$$ et $c = m$
- OW – CPA mais non IND – CPA : *textbook* RSA
- IND – CPA mais non NM – CPA : Elgamal ou Paillier

Chapitre III

Quelques applications

1. Chiffrement linéairement homomorphe

Définition III – 1 (Schéma de chiffrement asymétrique linéairement homomorphe)**.** Soit Π un schéma de chiffrement asymétrique, alors Π est dit **linéairement homomorphe**, si l'espace des messages clairs est un groupe $(\mathscr{M}, +)$, et si les deux algorithmes suivant existent :

- EvalAdd est un algorithme probabiliste polynomial qui prend en entrée la clef publique pk et deux chiffrés $c, c' \in \mathscr{C}$ de deux messages inconnus $m, m' \in \mathscr{M}$. Il ressort un élément c'' qui est un chiffré aléatoire de $m + m'$, obtenu avec la même loi de probabilité qu'en appliquant l'algorithme de chiffrement sur $m + m'$;
- EvalScal est un algorithme probabiliste polynomial qui prend en entrée la clef publique pk, un chiffré c d'un message inconnu $m \in \mathscr{M}$ et un entier $a \in \mathbf{N}$. Il ressort un élément c' qui est un chiffré aléatoire de $a.m := m + \cdots + m$ (a fois), obtenu avec la même loi de probabilité qu'en appliquant l'algorithme de chiffrement sur $k.m$.

Exemples :

1. Le schéma de Goldwasser Micali vu en TD : $\mathscr{M} = (\mathbf{Z}/2\mathbf{Z}, +)$. Soit $m, m' \in \mathscr{M}$, et leurs chiffrés : $c = g^m r^2$, $c' = g^{m'} r'^2$, avec $c, c' \in (\mathbf{Z}/n\mathbf{Z})^\times$. L'algorithme EvalAdd consiste à calculer $c'' := cc' r''^2$ avec $r'' \xleftarrow{\$} (\mathbf{Z}/n\mathbf{Z})^\times$. En effet, $c'' = g^{m+m'}(rr'r'')^2$ est un chiffré aléatoire de $m + m' \in \mathbf{Z}/2\mathbf{Z}$: si $m = m'$, c'' est un carré aléatoire donc un chiffré de 0, et si $m \neq m'$, ce n'est ni un carré modulo p ni modulo q, donc un chiffré de 1.
2. Le schéma de Paillier vu en TD : $\mathscr{M} = (\mathbf{Z}/n\mathbf{Z}, +)$. Soit $m, m' \in \mathscr{M}$, et leurs chiffrés : $c = (1+n)^m r^n$, $c' = (1+n)^{m'} r'^n$, avec $c, c' \in (\mathbf{Z}/n^2\mathbf{Z})^\times$. L'algorithme EvalAdd consiste à calculer $c'' := cc' r''^n$ avec $r'' \xleftarrow{\$} (\mathbf{Z}/n\mathbf{Z})^\times$. En effet, $c'' = (1+n)^{m+m'}(rr'r'')^n$ est un chiffré aléatoire de $m + m' \in \mathbf{Z}/n\mathbf{Z}$ car $(1+n)$ est d'ordre n modulo n^2.

   Pour l'algorithme EvalScal, étant donné c et a, on ressort $c^a r''^n = (1+n)^{ma}(r^a r'')^n$.
3. Le schéma d'Elgamal est aussi linéairement homomorphe mais vis à vis de la multiplication : par exemple si $(c_1, c_2) = (g^y, mh^y)$ et $(c'_1, c'_2) = (g^{y'}, m'h^{y'})$, $(c_1 c'_1 g^{y''}, c_2 c'_2 h^{y''}) = (g^{y+y'+h''}, mm' g^{y+y'+h''})$. On peut le rendre linéaire par rapport à l'addition, en chiffrant m par $(c_1, c_2) = (g^y, g^m h^y)$, mais dans ce cas, on doit se limiter à de petits m (et de faire un petit nombre d'additions), car le déchiffrement requiert de calculer le logarithme discret de g^m en base g pour retrouver m.

En 2009, avec les travaux de Gentry, est apparu le premier schéma de chiffrement dit totalement homomorphe complet (*Fully Homomorphic Encryption*, FHE), c'est à dire homomorphe à la fois pour l'addition et la multiplication, rendant possible l'évaluation de n'importe quelle fonction sur les chiffrés. Depuis, de nombreux efforts ont été menés pour rendre ce type de schéma utilisable en pratique. La sécurité des schémas actuels les plus performants est basée sur des problèmes algorithmiques utilisant les réseaux euclidiens (problème *(Ring) Learning With Errors*, (R)LWE, de Regev).

2. Vote électronique

Supposons que ℓ électeurs veuillent voter à un référendum. On va utiliser un chiffrement homomorphe additif, par exemple celui de Paillier. Une autorité a un couple (pk, sk) pour Paillier, la clef publique étant un module RSA, N. On suppose que $\ell < \mathrm{N}$ ce qui n'est pas une restriction en pratique (N fait au moins 2048 bits, soit plus de 600 chiffres décimaux).

On désigne par $m_i \in \{0,1\}$ le vote en clair de l'électeur i, avec 0 pour « non » et 1 pour « oui » (on suppose pour simplifier qu'il n'y a pas de vote blanc). Chaque électeur envoie c_i un chiffré de m_i avec la clef pk à l'autorité. Ces chiffrés sont publiés en ligne. À partir de ces chiffrés, il est possible pour chacun de calculer c un chiffré de $\sum_{i=1}^{\ell} m_i$ en utilisant l'algorithme EvalAdd ($c = \prod_{i=1}^{\ell} c_i$) pour Paillier.

En déchiffrant c avec sk, l'autorité retrouvera $\sum_{i=1}^{\ell} m_i \mod \mathrm{N} = \sum_{i=1}^{\ell} m_i$ dans $\mathbf{Z}$ car ce nombre est inférieur à $\ell < \mathrm{N}$. Ainsi on trouvera le résultat du vote : cela donne le nombre de votes pour « oui », donc si ce nombre est supérieur à $\ell/2$ le « oui » l'emporte.

Quelques Avantages et inconvénients de ce protocole :

- La sécurité sémantique du chiffrement assure qu'un adversaire extérieur ne pourra briser la confidentialité du vote, il ne sera pas distinguer les chiffrés de 0 de ce de 1.
- L'autorité, disposant de la clef privée sk, pourrait déchiffrer n'importe quel c_i au lieu de seulement c et retrouver le vote en clair de chaque électeur. On verra plus loin comment se protéger de cela.
- Un électeur malhonnête pourrait voter 10 ou −10, ce qui correspondrait à voter 10 fois. Pour remédier à cela, chaque électeur doit accompagner son vote chiffré c_i, d'une preuve à divulgation nulle de connaissance que c_i est un chiffré de 0 ou de 1.
- Comme le chiffrement utilisé est seulement IND – CPA, il n'y a pas de protection contre les attaques actives. L'adversaire pourrait modifier un c_i. Il faut donc protéger l'intégrité des chiffrés. Cela peut se faire par une signature numérique qui de plus permettrait d'authentifier les électeurs inscrits au vote.

On peut généraliser simplement ce protocole pour voter pour k candidats. Le vote en clair pour le candidat j, avec $0 \leqslant j \leqslant k-1$, sera A^j où $\mathrm{A} > \ell$ majore strictement le nombre d'électeurs ($\ell + 1$ suffit). La somme des votes décomposée en base A donnera $v_0 + v_1\mathrm{A} + \cdots + v_{k-1}\mathrm{A}^{k-1}$, où $v_j \leqslant \ell < \mathrm{A}$ est le nombre de votes pour le candidat j. Avec le protocole de Paillier, comme le déchiffrement retrouve ce nombre modulo N, pour avoir le résultat dans $\mathbf{Z}$ il faut que $\mathrm{A}^k < \mathrm{N}$. Donc si N fait 2048 bits, soit plus de 600 chiffres décimaux, on peut avoir 10^{10} personnes qui votent pour 60 candidats, donc couvrir une élection mondiale.

La résolution du problème de la confidentialité de chaque vote vis à vis de l'autorité, va se faire en partageant le déchiffrement entre plusieurs autorités, en utilisant une primitive appelée chiffrement à seuil, elle même basée sur le partage de secret.

3. Partage de secret

Un partage de secret est un protocole visant à partager un secret s entre n participants qui pourront collaborer pour reconstituer ce secret s. On peut faire une analogie avec un coffre fort contenant le secret, que l'on peut ouvrir en utilisant n clefs, chacune en possession d'une personne différente.

Un peu plus formellement, un partage de secret est constitué de deux algorithmes :

- Un algorithme de partage : Muni d'un secret s, un distributeur produit les parts, $s_1, \ldots, s_n$, et envoie la part s_i pour $i = 1, \ldots, n$ sur un canal sécurisé au participant i.
- Un algorithme de reconstruction : certains participants rendent publiques leur part et le secret s est reconstruit à partir de ces parts.

Un premier exemple : si s est composé de n bits, on note s_i le i–ème bit de s. Pour $i = 1, \dots, n$ le participant i reçoit s_i. Il est clair qu'avec toutes les parts on peut reconstruire s. Cependant, chaque participant apprend un bit de s, qui n'est plus complètement indéterminé : pour chaque participant, seul 2^{n-1} secrets sont possibles. Si t participants réunissent leur part, 2^{n-t} secrets sont possibles, avec $n-1$ parts, il n'y a plus que deux secrets possibles...

Il serait préférable que même avec $n-1$ parts, le secret soit complètement indéterminé. Cela implique que chaque part soit au moins aussi longue que le secret. On peut atteindre ceci avec le schéma suivant. Le secret s est un chaîne de k bits, vu comme élément de $\mathbf{F}_2^k$. On partage s en n parts comme suit : $s_1, s_2, \dots, s_{n-1}$ sont pris aléatoires dans F_2^k avec équiprobabilité, et on pose $s_n = s - \sum_{i=1}^{n-1} s_i$. Pour reconstruire, on fait la somme de toutes les parts, pour retrouver s. Par contre, quel que soit le secret s, si on prend $n-1$ parts (par exemple les $n-1$ premières), on obtient $n-1$ variables aléatoires uniformément distribuées dans F_2^k. Ainsi $n-1$ parts ne donnent aucune information sur s.

En pratique, il serait plus flexible de pouvoir reconstruire le secret avec moins de n parts. On parle de **partage de secret à seuil** de paramètre $[t, n]$ avec $t \leqslant n$ (t pour *threshold*) : avec $t-1$ parts parmi n on obtient aucune information sur le secret, et t parts permettent de reconstruire le secret. On peut construire un schéma $[t, n]$ en donnant à chaque sous-ensemble de t participants un partage $[t, t]$ du secret. Mais c'est évidemment peu efficace.

Le **schéma de Shamir** (1979) permet de construire un partage $[t, n]$ d'un secret s élément d'un corps fini $\mathbf{F}_q$. Il est basé sur l'interpolation de Lagrange et sur l'idée suivante : t points distincts d'évaluation permettent de définir de manière unique un polynôme de degré $t-1$, par contre avec seulement $t-1$ points, q polynômes sont possibles (correspondant aux différentes évaluations possibles en un autre point).

La procédure de partage procède ainsi. On considère fixés (et publics) n points distincts de $\mathbf{F}_q^*$, et on suppose $q > n$, $x_1, x_2, \dots, x_n$. Soit $s \in \mathbf{F}_q$ à partager. On prend $a_1, a_2, \dots, a_{t-1}$ aléatoires choisis avec équiprobabilité dans $\mathbf{F}_q$. On pose dans $\mathbf{F}_q[\mathrm{X}]$,

$$f(\mathrm{X}) = s + a_1\mathrm{X} + \cdots + a_{t-1}\mathrm{X}^{t-1}$$

de telle sorte que $f(0) = s$. Le participant i reçoit la part $s_i = f(x_i)$ pour $i = 1, \dots, n$.

La reconstruction utilise l'interpolation de Lagrange, à partir de t couples $(x_i, f(x_i))$ on reconstruit f de degré $t-1$ et on évalue en 0 pour retrouver s. Supposons, quitte à réordonner, que l'on reconstruit à partir des t premières parts, $s_1, \dots, s_t$, correspondant aux évaluations en $x_1, \dots, x_t$. Pour $1 \leqslant i \leqslant t$, le polynôme de Lagrange ℓ_i est

$$\ell_i(\mathrm{X}) = \prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{\mathrm{X} - x_j}{x_i - x_j}.$$

Ainsi, $\ell_i(x_j) = 0$ si $j \neq i$ et $\ell_i(x_i) = 1$. On pose $g(\mathrm{X}) = \sum_{j=1}^{t} s_j \ell_j(\mathrm{X})$, de telle sorte que $g(x_i) = s_i$ pour $1 \leqslant i \leqslant t$. Le polynôme $f - g$ est de degré au plus $t-1$ et à t racines (en les x_i) dans $\mathbf{F}_q$. Il est donc nul et $f = g$. On retrouve donc en $s = g(0)$.

Comme on veut juste la valeur en 0, on peut ne pas reconstituer complètement le polynôme g. On a $s = g(0) = \sum_{j=1}^{t} s_j \ell_j(0)$. En pré calculant les $\lambda_j := \ell_j(0) \in \mathbf{F}_q$, on trouve s comme combinaison linéaire des parts, $s = \sum_{j=1}^{t} \lambda_j s_j$.

Soit s un secret, montrons que $t-1$ parts ne donnent aucune information sur s. On peut supposer que ce sont les $s_i = f(x_i)$ pour $i = 1, \dots, t-1$. Soit la fonction $h_s : \mathbf{F}_q^{t-1} \to \mathbf{F}_q^{t-1}$ qui à $(a_1, \dots, a_{t-1})$, les valeurs aléatoires choisies lors du partage, associe $(s_1, \dots, s_{t-1})$. Cette fonction est une bijection : la réciproque est fournie par l'interpolation de Lagrange appliqué sur les couples $(0, s)$, (x_1, s_1), $\dots$, (x_{t-1}, s_{t-1}). Ainsi quelque soit s, $(s_1, \dots, s_{t-1})$ est uniformément distribué dans $\mathbf{F}_q^{t-1}$ et n'apporte donc aucune information sur s.

Soit s un secret et $(s_1, s_2, \dots, s_n)$ un partage par le protocole de Shamir. Le vecteur $(s, s_1, s_2, \dots, s_n)$ est un mot d'un code de Reed-Solomon. Ainsi le partage de secret de Shamir est linéaire : si $v = (s_1, \dots, s_n)$ (resp.

$v' = (s'_1, \dots, s'_n)$) est un partage de s (resp. de s'). Alors $v + v'$ est un partage de $s + s'$ et λv est un partage de λs pour tout $\lambda \in \mathbf{F}_q$.

Il existe en fait un parallèle entre codes linéaires et partages de secret linéaires, les codes MDS correspondant aux partages de secret à seuil.

Pour la reconstruction du secret, on a supposé que chaque participant était honnête, c'est à dire qu'il fournissait la bonne part s_i. S'il transmet une mauvaise valeur, alors un mauvais secret sera reconstruit, sans que cela soit détecté. Pour se protéger de telles attaques actives, on utilise la notion de **partage de secret vérifiable**. Une solution est que chaque participant prouve que sa part est légitime par exemple par une preuve d'appartenance à un langage (Chor, Goldwasser, Micali et Awerbuch, 1985).

Une solution plus pratique (Feldman 1987) est basée sur le problème du logarithme discret, en modifiant le protocole de Shamir. On suppose q premier et on note, G cyclique d'ordre q engendré par g (Feldman prenait un sous groupe de $(\mathbf{Z}/p\mathbf{Z})^\star$). Lors du partage, on publie $A_0 = g^s$, $A_1 = g^{a_1}, \dots, A_{t-1} = g^{a_t}$. Chaque utilisateur peut vérifier la cohérence de sa part par rapport à ces données publiques. En effet, le participant i peut calculer

$$\prod_{j=0}^{t-1} A_j^{x_i^j} = \prod_{j=0}^{t-1} g^{a_j x_i^j} = g^{\sum_{j=0}^{t-1} a_j x_i^j} = g^{f(x_i)}$$

et vérifier qu'il retrouve bien g^{s_i}.

Lors de la reconstruction, en utilisant le même calcul, chaque participant peut vérifier la part donnée par les autres participants avec le même type de calcul. Le secret ne sera reconstitué que si on obtient t parts valides.

4. Chiffrement à seuil