Cours de cryptologie avancée

Guilhem Castagnos

Septembre – Décembre 2025

version du 2 novembre 2025

Table des matières

Bibliographie et sources du cours

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

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

Dan Boneh et Victor Shoup *A Graduate Course in Applied Cryptography*, `https://toc.cryptobook.us`

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`

Ivan Damgård et Jesper Buus Nielsen, *Cryptologic Protocol Theory*, `https://cs.au.dk/~ivan/CPT.html`

Boaz Barak, *Crytography*, `https://www.cs.princeton.edu/courses/archive/fall07/cos433/`

Carmit Hazay et Yehuda Lindell, *Efficient Secure Two-Party Protocols*, Springer, 2010

Chapitre I

Sécurité CPA

1. Introduction à la sécurité réductionniste

Réductions

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. 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**.

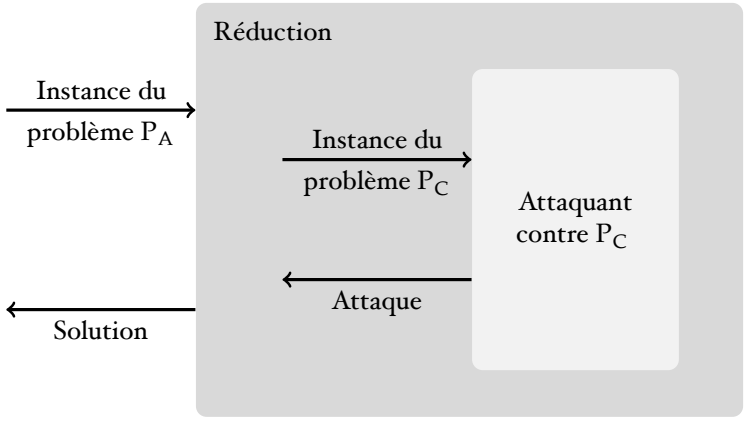


Schéma d'une preuve :

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.

Modélisation de la sécurité

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.

Pour modéliser l'attaquant, on utilise un **algorithme probabiliste polynomial**. En effet, l'aspect probabiliste est inhérent aux schémas cryptographiques : la clé privée est choisie aléatoirement, c'est par exemple une chaîne de k bits uniforme, et un adversaire polynomial peut toujours essayer de la deviner, ce qui donne une attaque avec probabilité de réussite $1/2^k$. En pratique, en prenant k assez grand, par exemple $k = 128$, cette probabilité de succès sera extrêmement faible. Ici, on s'intéressera à une sécurité asymptotique, en exprimant la probabilité de succès comme une fonction d'un paramètre de sécurité, un entier k. On introduit donc les fonctions négligeables.

Définition I – 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 $k \geqslant k_0$, $\nu(k) \leqslant \frac{1}{k^c}$.

La fonction $k \mapsto 1/2^k$ est négligeable. D'autres résultats utiles : les fonctions constantes $k \mapsto a$ avec $a > 0$ sont non négligeables, $\mathsf{negl}_1 + \mathsf{negl}_2$ est négligeable, $\mathsf{nonnegl} - \mathsf{negl}$ est non négligeable. 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. Remarquons que 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.

Les notions de sécurité que l'on verra sont dites **en moyenne**. On veut que le problème P_C de casser le schéma cryptographique soit dur étant donné une instance aléatoire correspondant à la génération de clef. Une sécurité définie uniquement dans le pire cas n'est pas utile en pratique. On utilisera donc des **variables aléatoires discrètes** pour modéliser toutes les entrées et sorties de l'attaquant et on étudiera la loi de probabilité (ou distribution) de sa sortie pour quantifier son succès. On définira P_C souvent en utilisant une expérience que l'on fait jouer à l'adversaire, écrite en pseudo-code, où il faudra interpréter toutes les variables comme des variables aléatoires (discrètes).

L'adversaire, $\mathscr{A}$, un algorithme polynomial probabiliste, 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 $f_{\mathscr{A}}$ (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 faits par $\mathscr{A}$. La sortie de $\mathscr{A}$, $f_{\mathscr{A}}(\mathrm{X}, \mathrm{R})$ est donc aussi une variable aléatoire.

Exemple. Modélisons plus précisément l'adversaire qui essaye de deviner la clef secrète. L'algorithme de génération de clef, étant donné le paramètre de sécurité k, génère donc $sk \xleftarrow{\$} \{0,1\}^k$, notation signifiant que sk est une variable aléatoire, suivant la loi discrète **uniforme** : on a équiprobabilité, $\Pr[sk = a] = 1/2^k$, pour tout $a \in \{0,1\}^k$. Il produit également une clef publique pk (variable aléatoire) fonction de sk. Soit $\mathscr{A}$

l'algorithme qui sur l'entrée pk, tire et retourne $\mathrm{R} \xleftarrow{\$} \{0,1\}^k$. Comme R est indépendante de sk (elle est choisie uniformément, indépendamment de toute autre variable), on a

$$\Pr[\mathrm{R} = sk] = \sum_{a\in\{0,1\}^k} \Pr[\mathrm{R} = a, sk = a] = \sum_{a\in\{0,1\}^k} \Pr[\mathrm{R} = a]\Pr[sk = a] = 2^k(1/2^k)^2 = 1/2^k.$$

La probabilité de succès de $\mathscr{A}$, $\Pr[\mathrm{R} = sk]$, est donc négligeable comme fonction de k.

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[1]. 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 définir l'inversion, on modélise comme vu plus haut : on requiert qu'un algorithme probabiliste polynomial $\mathscr{A}$ tentant d'inverser la fonction sur une image aléatoire n'a qu'une probabilité négligeable d'y arriver.

Définition I – 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,

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

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 (k en notation unaire (1111 ... 11 avec k symboles 1) pour permettre d'être polynomial en la taille de X même si $f(\mathrm{X})$ est beaucoup plus petit que X par exemple $|f(\mathrm{X})| = \log|\mathrm{X}|$.

Exemple. Considérerons f la fonction identité et calculons le succès de $\mathscr{A}$ étrangement, mais pédagogiquement défini ainsi : $\mathscr{A}(1^k, \mathrm{Y})$ tire un bit $b \xleftarrow{\$} \{0,1\}$, si $b = 0$, $\mathscr{A}$ retourne Y. Si $b = 1$, $\mathscr{A}$ retourne la chaîne de bit nulle de longueur k : 0 ... 0.

Si $b = 0$, le succès de A est 1, sinon c'est $1/2^k$. Donc, avec les probabilités totales, on trouve $1/2+1/2^{k+1} > 1/2$ constant donc non négligeable.

On peut de manière équivalente formuler le succès sous forme d'expérience (utile pour des notions de sécurité plus complexe).

$\mathbf{Exp}^{\mathsf{OW}}_{f,k}(\mathscr{A})$:

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

[1] Il est nécessaire, mais non suffisant, d'avoir P ≠ NP pour garantir l'existence de telles fonctions avec une sécurité en moyenne.

Remarque. 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 I – 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 $f(y) = f(x)$.

Exemple : la famille de fonctions à trappe RSA (1978). 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 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 objet de la définition suivante.

Définition I – 4 (hypothèse RSA)**.** 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 suivante :

$\mathbf{Exp}^{\mathsf{OW}}_{\mathsf{GenRSA},k}(\mathscr{A})$:

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.

Cette hypothèse est plus forte que de supposer que la factorisation des entiers RSA est un problème dur (si on sait factoriser N on peut retrouver la trappe d et inverser la fonction $x \mapsto x^e$). Montrer qu'elle est strictement plus forte ou équivalente est un problème ouvert.

3. Schéma de chiffrement asymétrique

Définition I – 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 d'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 $1 - \mathsf{negl}(k)$. 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. Le système RSA est donc sûr au sens TB – CPA sous l'hypothèse que la factorisation est difficile.

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

Définition I – 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

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

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 I – 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 $\mathrm{R} \subset \mathscr{M} \times \mathscr{Z}$ et $\mathrm{R}(m, z)$. Par exemple la relation peut être une relation d'égalité, $z = m$: l'adversaire doit retrouver m en entier, on retrouve la notion de sens-unique ; cela peut être le poids de Hamming de m ; $z = \mathrm{W_H}(m)$; le premier bit de m...

Un exemple pour motiver la définition : 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 apporte donc une information nouvelle.

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 I – 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, $\mathbf{Exp}_{\Pi,k}^{\mathsf{IND-CPA}}(\mathscr{A})$:

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

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

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,

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

Remarques :

1. L'état d'information 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 $\mathbf{Adv}_{\Pi,k}^{\mathsf{IND-CPA}}(\mathscr{A}) = 0$.
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 :

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

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

Idée de la preuve. La définition de la sécurité sémantique est dite une définition par simulation (on exhibe un simulateur mimant la réponse de l'attaquant) alors que celle de l'indistinguabilité est dite *game-based* : on fait jouer une expérience à l'attaquant. Cette définition de sécurité sémantique correspond au paradigme monde réel / monde idéal utilisé pour définir la sécurité des protocoles de calcul multipartite (MPC). Dans le monde réel, l'attaquant a accès à (c, α, pk). Dans le monde idéal on voudrait que l'information transite « de manière magique » directement d'Alice à Bob et le simulateur n'a accès qu'à (α, pk).

Dans le sens IND–CPA ⇒ sémantiquement sûr CPA, on veut montrer que si le chiffrement est indistinguable alors on peut construire un simulateur tel que le monde réel et le monde idéal sont indistinguables. La technique usuelle est de construire le simulateur en déplaçant l'adversaire dans le monde idéal : on construit son entrée complète sans connaître le secret (ici m) et on veut montrer que sa sortie reste distribuée de manière indistinguable. Ici, cela veut dire construire un chiffré d'un message m' différent de m (on peut prendre la chaîne de bits 00 ... 00) et on veut que la sortie de $\mathscr{A}$ reste indistinguable en ayant un chiffré de m ou un chiffré de m' : ceci sera assuré par le fait que le chiffrement est IND – CPA.

L'autre sens est plus direct, l'indistinguabilité étant un cas particulier de la sécurité sémantique : l'attaquant contre l'indistinguabilité 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 ».

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 (en utilisant $\mathscr{A}$ en simulant le chiffré auquel il a accès) :

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 $\mathscr{Enc}$ (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. $(m, \mathrm{R}, \alpha) \leftarrow \mathscr{Enc}(pk)$
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 $\mathsf{IND-CPA}$ à partir de $\mathscr{A}$.

On définit $\mathscr{B}_1$:

1. $\mathscr{B}_1$ reçoit pk
2. Il obtient (m, R, α) de $\mathscr{Enc}(pk)$
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 $\mathsf{CPA} \Rightarrow \mathsf{IND-CPA}$.

On suppose donc que Π est sémantiquement sûr CPA et qu'il n'est pas $\mathsf{IND-CPA}$. Soit $\mathscr{B} = (\mathscr{B}_1, \mathscr{B}_2)$ un attaquant $\mathsf{IND-CPA}$ ayant un avantage non négligeable. On peut supposer que les messages m_0, m_1 choisis par $\mathscr{B}_1$ sont distincts. Comme dit plus haut, 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 ». Plus précisément, on construit $\mathscr{Enc}$ 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 $\mathrm{R}(m, z) \Leftrightarrow m = m_z$ avec $z \in \{0,1\}$.
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 $z := b$

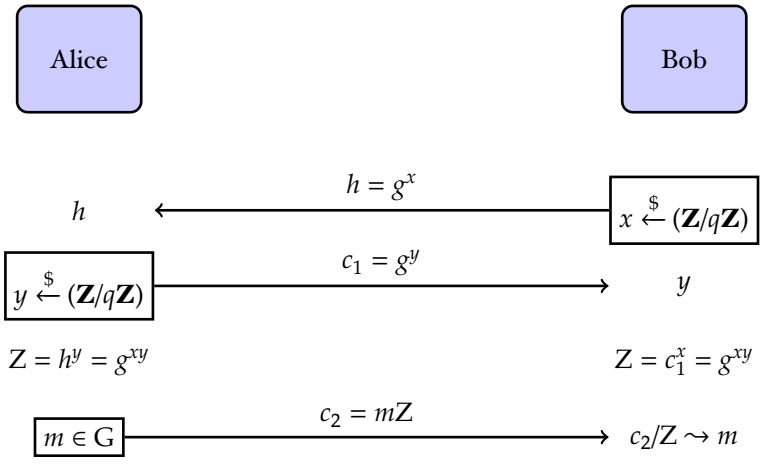
Il est clair que la probabilité $\Pr\left(\mathbf{Exp}^{\mathsf{sem-sec-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) = \Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{B}) = 1\right)$. Par contre tout algorithme $\mathscr{S}$ qui n'aurait accès qu'à (α, pk) n'aurait aucune information sur le bit $b^\star$ tiré par $\mathscr{E}nv$ ($b^\star$ est déterminé après α). C'est à dire que l'entrée de $\mathscr{S}$ et donc sa sortie est indépendante de $b^\star$. On utilise maintenant le fait que l'on a deux variables aléatoires à valeurs dans $\{0,1\}$, indépendantes et que l'une est uniforme. Elles sont donc égales avec probabilité $1/2$, c'est à dire que $\mathscr{S}$ a une chance sur deux de retourner le bon bit, quelle que soit sa stratégie. 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. □

Remarque Ce théorème nous permettra dans la suite de montrer qu'un chiffrement est sémantiquement sûr en utilisant la notion d'indistinguabilité bien plus adaptée aux preuves.

Exemple Elgamal (1985) : Soit GenDL 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 GenDL puis choisit x uniforme dans $\mathbf{Z}/q\mathbf{Z}$ et calcule $h = g^x$. KeyGen retourne $pk = (\mathbf{G}, q, g, h)$ et $sk = x$. On pose $\mathscr{M} = \mathbf{G}$ et $\mathscr{C} = \mathbf{G}\times\mathbf{G}$. 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.

La sécurité d'Elgamal est donc liée à celle de l'échange de clef Diffie-Hellman. On va montrer que casser la notion de sens-unique d'Elgamal revient à retrouver $Z = g^{xy}$ étant donné $X = g^x$ et $Y = g^y$, c'est à dire former le triplet Diffie-Hellman (g^x, g^y, g^{xy}) : si on arrive à retrouver m, on peut calculer Z. De même, on va voir que casser l'indistinguabilité revient à distinguer un triplet Diffie-Hellman d'un triplet uniforme.

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

Définition I – 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 $X = g^x$ et $Y = g^y$
3. $\mathscr{A}$ prend $G, q, g, (X, Y)$ en entrée et renvoie $Z \in G$
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 I – 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.

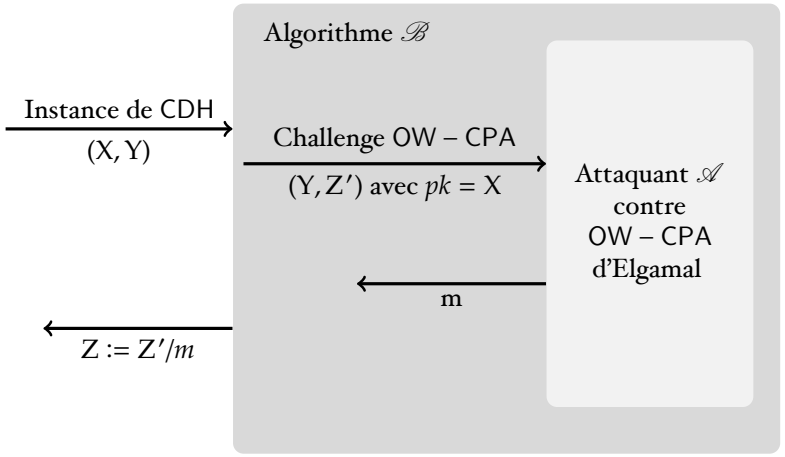
On conjecture que cette hypothèse DDH est vraie dans les sous groupes de $(\mathbf{Z}/p\mathbf{Z})^\times$ d'ordre q avec q un grand diviseur premier de $p-1$ avec p premier (par exemple en utilisant des nombres premiers de Sophie Germain : $p = 2q+1$, on travaille dans ce cas avec les carrés de $(\mathbf{Z}/p\mathbf{Z})^\times$). 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 $e(\mathrm{P}, c\mathrm{P}) = e(a\mathrm{P}, b\mathrm{P})$ si et seulement si $c = ab$ modulo l'ordre de P).

Théorème I – 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 $pk = (\mathbf{G}, q, g, X)$ 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 $c^\star = (Y, m_{b^\star} Z)$
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 $b^\star$ (*one-time pad*). Le chiffré $c^\star$, en particulier, et la vue de $\mathscr{A}$ c'est à dire ses entrées sont donc indépendantes du bit $b^\star$. Il en est de même de sa sortie b. Comme déjà vu, comme $b^\star$ est uniforme, même un adversaire avec une puissance de calcul infini ne peut donc deviner le bit $b^\star$ qu'avec probabilité 1/2 quelle que soit sa stratégie. 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 calculatoirement 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 » : les deux jeux procèdent similairement à moins que l'évènement F_i (resp. F_{i+1}) se produise dans le jeu i (resp. dans le jeu $i+1$) c'est à dire que $\Pr(S_i \wedge \neg F_i) = \Pr(S_{i+1} \wedge \neg F_{i+1})$ et on suppose que $\Pr(F_i) = \Pr(F_{i+1}) := \epsilon$. On voit alors facilement que $|\Pr(S_i) - \Pr(S_{i+1})| \leqslant \epsilon$:

$$\begin{aligned} |\Pr(S_i) - \Pr(S_{i+1})| &= |\Pr(S_i \wedge F_i) + \Pr(S_i \wedge \neg F_i) - \Pr(S_{i+1} \wedge F_{i+1}) - \Pr(S_{i+1} \wedge \neg F_{i+1})| \\ &= |\Pr(S_i \wedge F_i) - \Pr(S_{i+1} \wedge F_{i+1})| \\ &\leqslant \epsilon \end{aligned} \tag{I.1}$$

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

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 plus loin. 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 ⇒ IND – CPA. Dans le cas d'attaque actives adaptatives, on a une équivalence entre les deux notions.

7. Résumé

On a

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

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 II

Preuves interactives à divulgation nulle de connaissance

1. Introduction

Une preuve mathématique permet de convaincre quelqu'un d'un énoncé. On peut aussi montrer qu'un objet a certaines propriétés. Par exemple, Alice peut montrer à Bob qu'un entier est premier en utilisant un certificat produit par un test de primalité, ou plus facilement montrer qu'un entier n'est pas premier : 1457 en donnant à Bob une factorisation non triviale : 31×47.

Cependant, Bob apprend ici de l'information au-delà que 1457 n'est pas premier : sa factorisation. Dans une preuve à divulgation nulle de connaissance (*Zero-Knowledge proof*, ou *ZK proof*), Alice va montrer à Bob qu'une assertion est vraie, Bob va être convaincu que c'est le cas, mais il ne va pas apprendre d'information supplémentaire.

Ces preuves ont été introduite par Goldwasser, Micali et Rackoff en 1982-85. Elles ont de nombreuses applications en cryptographie.

Quelques applications en cryptographie

Les preuves ZK permettent de construire des **protocoles d'authentification** : Alice prouve son identité à Bob (le serveur d'authentification) en prouvant qu'elle connaît un secret. Avec l'authentification par mot de passe, Bob (ou un tiers ayant accès au serveur d'authentification) apprend une information (le mot de passe) au-delà qu'Alice est la personne connaissant le mot de passe. Avec une preuve ZK, Bob a par exemple une instance d'un problème algorithmique difficile (un nombre RSA N, une instance du problème du logarithme discret, $h = g^x$), Alice a une solution (p,q tels que $N = pq$, x tel que $h = g^x$). Pour s'authentifier, Alice prouve en ZK à Bob qu'elle connaît cette solution, et Bob n'apprend aucune information supplémentaire.

On verra que les preuves ZK permettent aussi de produire des signatures numériques.

Les preuves ZK ont aussi de nombreuses applications dans le contexte d'un protocole de **calcul multipartite sécurisé** (*secure multi-party computation*, MPC). Supposons par exemple qu'Alice et Bob veulent faire un calcul, pour déterminer si $a > b$ mais en gardant leurs entrées a et b secrètes. Ils vont exécuter un protocole. Si on sait qu'Alice et Bob suivent le protocole à la lettre (modèle de sécurité honnête mais curieux), on peut construire un protocole relativement simple, masquant la valeur de a à Bob et celle de b à Alice. Mais si Alice triche dans le protocole, elle peut fausser le résultat ou essayer d'avoir de l'information sur b. Une solution souvent utilisée pour obtenir des protocoles sûrs contre des adversaires malveillants est que tout le long du protocole Alice et Bob montrent qu'ils suivent bien toutes les étapes, en utilisant des preuves ZK pour montrer qu'ils envoient bien les données telles que spécifiées par le protocole, sans fournir d'information supplémentaire à l'autre partie.

2. Preuves interactives à divulgation nulle de connaissance

Les preuves ZK sont des protocoles interactifs entre deux participants. On peut les modéliser en utilisant deux machines de Turing interactives, c'est-à-dire des machines de Turing probabilistes qui sont en outre équipées de bandes de communication permettant à une machine d'envoyer et de recevoir des messages de l'autre.

Dans une **preuve interactive**, un participant, appelé le prouveur, P, a une puissance de calcul infinie, et l'autre, appelé le vérificateur, V, est polynomial. Le but est pour P de convaincre V qu'une chaîne de bits x, un énoncé, appartient ou non à un langage binaire L (un sous-ensemble de $\{0,1\}^*$). Les deux participants ont en entrée x. À la fin du protocole, V ressort accept ou reject.

Définition II – 1. La paire (P, V) est une **preuve interactive** pour un langage binaire L si elle satisfait les deux conditions suivantes :

- *Completeness* (complétude/consistance) : Si $x \in$ L, alors la probabilité que (P, V) rejette x est négligeable (en la longueur de x) ;
- *Soundness* (correction ou validité ou robustesse) : Si $x \notin$ L alors pour tout prouveur P*, la probabilité que (P*, V) accepte x (**erreur de soundness**) est négligeable (en la longueur de x).

Remarques :

- Dans la définition de soundness, « Pour tout prouveur P* », signifie « quelque soit la stratégie du prouveur » : il peut « tricher » et ne pas suivre le protocole original (c'est-à-dire la description originale de (P, V)).
- S'il n'y a pas d'interaction et que V est déterministe et que l'erreur de soundness est nulle, alors on retrouve la classe NP. En fait, on a NP $\subset$ PSPACE = IP. Cette dernière égalité a été montrée par Shamir (1992). La classe PSPACE représente les langages qui peuvent être vérifiés avec un algorithme polynomial en mémoire mais pas forcément en temps.
- Dans la définition, P, P* peuvent être non polynomiaux (en temps). En restreignant à des algorithmes polynomiaux, on obtient des **arguments interactifs**. Il faut pour cela que P ait accès à une donnée privée (un secret, un témoin d'appartenance au langage), qui lui permette de convaincre V. Cette restriction permet de montrer la *soundness* sous une hypothèse algorithmique.

Exemple, le protocole QR

Soit N un entier RSA. On considère le langage des carrés modulo N. On note sous forme de relation :

$$\mathrm{QR} = \left\{ \left((\mathrm{N}, x);\, w \right) \;\middle|\; \mathrm{N} \text{ entier RSA}, x \in (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times, x = w^2 \pmod{\mathrm{N}} \right\},$$

où (N, x) sera un énoncé à prouver et w un témoin d'appartenance au langage.

Le but est d'obtenir un protocole qui ne donne pas d'information sur w (pour l'instant la définition de preuve interactive ne fait pas intervenir cette notion, mais on va voir cela plus loin). Une première idée de protocole est la suivante. Le prouveur envoie $y = u^2$ au vérifieur. S'il sait donner u et uw à V, celui-ci peut vérifier que $u^2 = y$ et $(uw)^2 = xy$ et être convaincu que x est un carré car y l'est (on utilise que l'ensemble des carrés est un groupe). Mais cela ne sera pas sans divulgation : V apprend w. On choisit donc que le prouveur n'envoie aléatoirement que l'un des deux. Cela donne le protocole de la figure II.1.

Vérifions que l'on a bien une preuve interactive.

Completeness : si x est un carré modulo N avec $x = w^2$, et que le prouveur suit le protocole, alors V accepte bien car $z^2 = u^2(w^2)^b = yx^b$.

Soundness : on suppose que x n'est pas un carré, et on considère un prouveur malveillant P* (qui ne suit pas forcément le protocole) et on analyse la probabilité que V accepte. On note y et z les deux messages envoyés par P*. On peut supposer que $y \in (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$ (V peut le vérifier).

Entrée : N, x

- P $\rightarrow$ V : P choisit $u \xleftarrow{\$} (\mathbf{Z}/N\mathbf{Z})^\times$ et envoie $y = u^2$ à V
- P $\leftarrow$ V : V choisit $b \xleftarrow{\$} \{0,1\}$ et l'envoie à P
- P $\rightarrow$ V : P envoie $z = uw^b$ à V

Sortie : V accepte la preuve si et seulement si $z^2 = yx^b$.

Fig. II.1 : Protocole pour QR

Si y est un carré, on écrit $y = u^2$. Si $b = 1$, V ne peut pas accepter car sinon $z^2 = u^2x$, et donc $x = (zu^{-1})^2$ et on a une contradiction. Comme $b = 1$ avec probabilité 1/2, V accepte la preuve avec probabilité $\leqslant 1/2$. De même, si y n'est pas un carré et $b = 0$, V ne peut accepter sinon $z^2 = y$. De même V accepte la preuve avec probabilité $\leqslant 1/2$.

Ainsi V accepte dans tous les cas un mauvais énoncé avec probabilité $p \leqslant 1/2$. Pour avoir une erreur de *soundness* négligeable, on répète k fois ce protocole, de manière indépendante, avec $|N| = k$ et V accepte si et seulement si les k vérifications sont valides. La probabilité d'accepter un mauvais énoncé est $p^k \leqslant 1/2^k$, qui devient négligeable (erreur de *soundness*).

Zero-Knowledge

On veut maintenant définir le fait qu'un vérificateur V^* n'apprend rien de supplémentaire de l'exécution du protocole avec un prouveur honnête, en dehors que l'énoncé est vrai. Ceci même si V^* ne suit pas le protocole.

Pour définir formellement l'aspect *zero-knowledge*, on utilise, comme pour la sécurité sémantique, un simulateur.

Définition II – 2. Une preuve interactive (ou un argument interactif) (P, V) pour un langage binaire L est **à divulgation nulle de connaissance** ou ***zero-knowledge*** (ZK) si pour tout vérificateur probabiliste polynomial V^*, il existe un algorithme probabiliste polynomial (en moyenne) qui produit de manière indistinguable les communications entre P (honnête) et V^* sur l'entrée $x \in L$.

Notons que cette définition ne contredit pas la notion de *soundness*. Le simulateur n'est pas tout puissant comme P ou ne connaît pas son secret mais il possède un avantage sur celui-ci pour simuler la conversation : il peut générer les messages dans n'importe quel ordre alors que le prouveur doit suivre l'ordre du protocole.

On peut être plus précis sur le « indistinguable » de la définition et définir plusieurs variantes de *zero-knowledge* (parfait, statistique, calculatoire).

Exemple, retour sur le protocole QR

Supposons que V soit honnête. Alors le bit qu'il choisit est uniforme. On peut construire un simulateur en choisissant d'abord b et z uniformément, puis en posant $y = z^2x^{-b}$. Le triplet (y, b, z) a alors la même distribution qu'une conversation entre P et V sur un carré x. En effet, y est bien un carré uniformément distribué (que b soit nul ou non), b est uniforme et comme $z^2 = yx^b$, z est l'une des racines carrées (choisie uniformément comme z est uniforme) de yx^b. On a montré ici une propriété ZK pour un vérificateur honnête (HVZK).

Considérons maintenant un V^* arbitraire comme dans la définition de *zero-knowledge*. La difficulté est que V^* peut choisir b non pas uniformément mais suivant la valeur de y, le premier message de P. Mais comme le simulateur peut dépendre de V^*, il peut utiliser sa stratégie. On définit le simulateur dans la figure II.2.

Si le simulateur termine, il sort bien une conversation distribuée de manière identique à celle du protocole : comme précédemment y est un carré uniforme donc $b \leftarrow V^*(y)$ est également distribué comme dans le protocole ainsi que z.

Entrée : N, x avec x un carré modulo N

1. Choisir $b' \xleftarrow{\$} \{0,1\}$
2. Choisir $z \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$
3. Calculer $y := z^2 x^{-b'}$
4. Invoquer V^* sur le message y pour obtenir un bit b
5. Si $b = b'$, retourner (y, b, z). Sinon, repartir au point 1

Fig. II.2 : Simulateur ZK pour QR

Maintenant montrons qu'en moyenne le simulateur termine en temps polynomial. Comme y est un carré uniforme quelle que soit la valeur de b', y est indépendant de b', et donc b, la sortie de $\mathrm{V}^*(y)$, est également indépendant de b'. Comme b' est uniforme on a $\Pr[b = b'] = 1/2$. Donc, en moyenne en deux itérations le simulateur s'arrête (espérance égale à $1/p$ de la loi géométrique de paramètre $p = 1/2$: nombre d'épreuves de Bernoulli indépendantes de probabilité de succès p nécessaires pour obtenir le premier succès).

On a écrit un simulateur pour une itération. Pour avoir une erreur de *soundness* négligeable, on a vu qu'il fallait répéter k fois où k est la taille de N. On admet que tout protocole *zero-knowledge* répété indépendamment et séquentiellement un nombre polynomial de fois reste *zero-knowledge*.

Preuves de connaissance

Les preuves que l'on a défini jusqu'à présent sont des preuves d'appartenance à un langage : elles permettent au prouveur P de convaincre le vérificateur V que l'énoncé x est dans un certain langage. Les preuves de connaissance vont plus loin, elles assurent V que P connaît un témoin d'appartenance au langage. Sur l'exemple du protocole QR, cela signifie que P connaît une racine carrée w du carré x modulo N.

On parle de preuves de connaissance interactives à divulgation nulle de connaissance (*Zero-Knowledge Proof of Knowledge*, ZKPoK). On peut également définir des arguments, ZKAoK, lorsque le prouveur est polynomial.

Pour définir de telles preuves de connaissance, on remplace la notion de *soundness* par une notion plus forte de *knowledge soundness* : il est possible d'interagir, en temps polynomial, avec un prouveur P^* qui valide la preuve avec une probabilité plus grande que l'erreur de *soundness* pour calculer le témoin. L'idée est que si P^* valide la preuve, alors il doit connaître le témoin. L'interaction se fait avec un algorithme, appelé extracteur, qui peut en particulier « rembobiner » P^*, c'est-à-dire revenir à un certain point du protocole sans le recommencer depuis le début (ainsi l'extracteur a plus de pouvoir qu'un vérifieur, et on ne contredit pas la notion de *zero-knowledge*).

Définition II – 3. On dit que (P, V) est une **preuve interactive de connaissance** pour un langage L si c'est une preuve interactive pour L, où l'on remplace la propriété de *soundness* par la propriété de ***knowledge soundness*** avec erreur κ : Pour tout énoncé x et prouveur P^* tel que la probabilité p que $(\mathrm{P}^*, \mathrm{V})$ accepte x est strictement supérieure à δ, c'est-à-dire $p = \kappa + \rho$ avec $\rho > 0$, alors il existe un algorithme avec accès sous forme d'oracle à P^* qui sur l'entrée x ressort un témoin w d'appartenance de x à L en fonctionnant en temps moyen inférieur à $poly(|x|)/\rho$.

Cette définition implique celle de *soundness* si l'erreur κ est négligeable : une preuve de connaissance est donc aussi une preuve interactive. En effet, si $x \notin \mathrm{L}$ et $(\mathrm{P}^*, \mathrm{V})$ accepte avec probabilité non négligeable, alors l'extracteur permet de retrouver un témoin d'appartenance x à L et on a une contradiction.

Exemple, nouveau retour sur le protocole QR

L'idée est la suivante : si P^* est capable de valider la preuve pour les deux valeurs du challenge b, pour un même engagement y, alors on peut retrouver w à partir de ces deux réponses. En effet,si z désigne la réponse pour $b = 0$ et z' celle pour $b = 1$, alors $z^2 = y$ et $z'^2 = yx$. Donc $w = z'z^{-1}$ est une racine de x et donne un témoin d'appartenance de x au langage des carrés modulo N.

On définit donc l'extracteur suivant :

1. Sur l'entrée (N, x), invoquer le prouveur P^* qui renvoie y
2. Envoyer le bit $b = 0$ à P^*, qui renvoie z
3. Rembobiner P^*, après l'envoi de y
4. Envoyer le bit $b = 1$ à P^*, qui renvoie z'
5. Si $z^2 = y$ et $z'^2 = yx$, retourner $z'z^{-1}$. Sinon, repartir au point 1.

Si l'extracteur termine, il renvoie bien un témoin comme vu plus haut. On cherche donc à déterminer le temps d'exécution moyen de l'extracteur. On pose $\kappa = 1/2$, puisque sur une itération un prouveur P^* peut toujours faire accepter V avec probabilité 1/2 en devinant le bit b (c'est-à-dire de manière similaire au simulateur HVZK). Supposons donc que P^* ait une probabilité $p = 1/2 + \rho$ avec $\rho > 0$ de faire accepter V sur l'énoncé x.

Montrons qu'avec probabilité au moins 2ρ, P^* renvoie à l'étape 1 un y tel qu'il retourne des z et z' valides respectivement pour $b = 0$ et $b = 1$. La preuve de ce fait est générique et s'applique à toute preuve ZK partageant la même structure (un engagement du prouveur, puis un défi du vérificateur avec un $b \in \{0,1\}$ et une réponse du prouveur).

Pour dénombrer les possibilités, on considère une matrice dont les ℓ lignes sont indexées par les choix d'aléas de P^* et les colonnes par la valeur de b. On met un 1 dans la matrice si avec ce choix d'aléa (donc de y) et de défi b, P^* retourne un z qui est accepté, et un 0 sinon. On a donc $A := 2\ell p = 2\ell(1/2 + \rho) = \ell + 2\rho\ell$ entrées 1 dans la matrice. On note m le nombre de lignes de la forme $(1,1)$: ces lignes correspondent à un choix de y qui donne une conversation acceptée pour $b = 0$ et $b = 1$. On a au maximum un 1 sur chaque ligne et m supplémentaires venant des lignes $(1,1)$, donc $A = \ell + 2\rho\ell \leqslant \ell + m$ et donc $2\rho\ell \leqslant m$.

Ainsi, avec probabilité $m/\ell \geqslant 2\rho$, le y choisi par le prouveur va permettre d'extraire. L'extracteur doit donc lancer le prouveur en moyenne $1/(2\rho)$ fois pour avoir le bon y et on a la complexité attendue.

Mise en gage et ZKP pour tout langage de NP

Un protocole de mise en gage ou *commitment scheme* est l'analogue d'une enveloppe scellée. Il permet de fixer une valeur et de la cacher temporairement. Plus formellement, on a la définition suivante.

Définition II – 4. Un **protocole de mise en gage** est la donnée de deux algorithmes :

- $\mathsf{Setup}(1^k)$ produit des paramètres publics pp ;
- $\mathsf{Com}_{pp}(m, r)$ est un algorithme déterministe qui sous l'entrée $m \in \mathscr{M}$ et l'aléa $r \in \mathscr{R}$ produit une mise en gage $c \in \mathscr{C}$ de m.

L'utilisation typique est la suivante : Alice et Bob se mettent d'accord sur des paramètres publics pp produit par $\mathsf{Setup}(1^k)$. Alice met en gage m avec un aléa r et fourni le c produit par $\mathsf{Com}_{pp}(m, r)$ à Bob. Plus tard, dans une phase d'ouverture de c, elle révèle (m, r) à Bob qui peut vérifier que $c = \mathsf{Com}_{pp}(m, r)$.

On veut deux propriétés de sécurité. Après la mise en gage, Bob n'obtient aucune information sur m à partir de c (***hiding***). Pour cela on définit un jeu similaire à l'expérience $\mathsf{IND} - \mathsf{CPA}$. D'autre part, Alice ne doit pouvoir convaincre Bob que c engage une valeur m' différente de m (***binding***), c'est à dire qu'une fois fixé $c = \mathsf{Com}_{pk}(m, r)$, il est difficile de trouver (m', r') tel que $m \neq m'$ et $c = \mathsf{Com}_{pk}(m', r')$.

On peut construire un protocole de mise en gage avec un chiffrement IND-CPA : on aura la propriété de *hiding* sous une hypothèse et la propriété de *binding* de manière inconditionnelle : pour une clef publique et un chiffré de m, si on pouvait ouvrir c sur une autre valeur, le chiffrement ne serait pas correct. On verra

en TD qu'il est possible de construire plus directement des protocoles de mises en gage. On peut également montrer que l'existence des fonctions à sens unique implique celle de commitments (Naor, 1989).

Les commitments ont de nombreuses applications dans des protocoles interactifs. Par exemple, si Alice et Bob doivent s'échanger des valeurs a et b et que l'on veut garantir que Bob ne choisit pas b en fonction de a ou l'inverse : Alice va envoyer une mise en gage de a, Bob va lui envoyer b puis Alice va ouvrir sa mise en gage. Dans le domaine des preuves ZK, on a l'application suivante.

Théorème II – 5 (Goldreich-Micali-Wigderson 1986)**.** Si les commitments inconditionnellement *binding* existent alors tout langage dans NP admet une preuve ZK.

La preuve du théorème consiste à donner une preuve ZK pour un langage NP complet (voir TD). Une mise en gage va permettre au prouveur d'envoyer un premier message au vérifieur sans lui donner d'information, ce qui va aider à garantir la propriété de *zero-knowledge* par la propriété d'*hiding*. Le *binding* va lui intervenir pour la *soundness*.

Ce résultat a été étendu à tous les langages admettant une preuve interactive (Ben-or et al., 1988). D'autre part, ce résultat assure qu'il est possible de fournir une ZKP pour le problème SAT de satisfaisabilité booléenne. Étant donné une instance x d'un langage $\mathrm{L} \in \mathrm{NP}$, on peut donc construire un circuit booléen à partir de x et utiliser cette ZKP pour prouver que $x \in \mathrm{L}$. Ceci peut donner des preuves coûteuses, notamment en communication. Depuis les années 2010, des méthodologies ont été développées pour obtenir des preuves « succinctes » (dont les communications sont beaucoup plus petites que l'énoncé) et rapides à vérifier pour prouver des énoncés complexes comme « je connais un m tel que $\mathrm{SHA}-2(m) = 0 \ldots 0$ ».

3. *Sigma protocol*

Exemple du protocole de Schnorr

On souhaite prouver la connaissance d'un logarithme discret. Soit G un groupe cyclique d'ordre fini premier q de k bits, g un générateur. On suppose qu'il est possible de vérifier efficacement qu'un élément est dans G. On considère le protocole de la figure II.3 pour la relation

$$\mathrm{DL} = \{((g,h);\, w) \;\mid\; h = g^w\}.$$

Entrée : g, h

- $\mathrm{P} \to \mathrm{V}$: P choisit un entier $r \xleftarrow{\$} \mathbf{Z}/q\mathbf{Z}$ et envoie $t = g^r$ à V
- $\mathrm{P} \leftarrow \mathrm{V}$: V choisit $c \xleftarrow{\$} \mathscr{C} = \mathbf{Z}/q\mathbf{Z}$ et l'envoie à P
- $\mathrm{P} \to \mathrm{V}$: P répond $z = r + wc$ modulo q

Sortie : V accepte la preuve si et seulement si $g^z = th^c$.

Fig. II.3 : Protocole de Schnorr pour DL (1991)

Montrons les propriétés de ce protocole. Si $h = g^w$ et P et V suivent le protocole, alors on a bien $g^z = g^{r+wc} = th^c$ et V accepte toujours, on a la propriété de *completeness*.

Étudions la propriété de *knowledge soundness*. Supposons que l'on ait une première conversation acceptée (t, c, z) et que l'on puisse rembobiner le prouveur après l'envoi de t et obtenir une autre conversation acceptée, (t, c', z') pour $c \neq c'$. On a alors $g^z = th^c$, $g^{z'} = th^{c'}$ et donc $g^{z-z'} = h^{c-c'}$, comme $c - c'$ est inversible modulo q, en supposant de plus que V a vérifié que h était un élément de G, on obtient que $h = g^{(z-z')(c-c')^{-1}}$. Donc

$$w = (z - z')(c - c')^{-1}.$$

On peut donc extraire le témoin. On va voir plus loin qu'il est possible d'obtenir ces deux conversations acceptées pour un prouveur qui fait valider la preuve avec probabilité $> 1/|\mathscr{C}|$.

Pour la propriété de *zero-knowledge*, supposons que V est honnête. On peut alors simuler une conversation acceptée, en posant $z \xleftarrow{\$} \mathbf{Z}/q\mathbf{Z}$, $c \xleftarrow{\$} \mathscr{C}$, et $t = g^z h^{-c}$. On a bien que (t, c, z) a la même distribution qu'une conversation acceptée par V : $t \xleftarrow{\$} \mathrm{G}$, $c \xleftarrow{\$} \mathscr{C}$ et z est l'unique entier modulo q qui valide l'équation de vérification. On a une propriété de HVZK. Par contre, si V^* est malhonnête, on ne sait pas prouver que ce protocole est *zero-knowledge* (ou non). La technique vue pour QR consistant à écrire un simulateur devinant la valeur du challenge utilisé par V^* ne fonctionne plus car il y a un choix exponentiel en k pour le challenge.

Sigma Protocol (Cramer 1997)

De nombreuses preuves *zero-knowledge* partagent la même structure en 3 passes avec un engagement, un défi et une réponse et où l'on peut extraire un témoin à partir de deux conversations acceptées (c'est le cas pour le protocole QR vu précédemment avec $\mathscr{C} = \{0, 1\}$).

Entrée : x

- $\mathrm{P} \to \mathrm{V}$: P envoie un engagement t à V
- $\mathrm{P} \leftarrow \mathrm{V}$: V choisit un défi $c \xleftarrow{\$} \mathscr{C}$ et l'envoie à P
- $\mathrm{P} \to \mathrm{V}$: P répond z

Sortie : V renvoie accept ou reject à partir de (x, t, c, z)

Fig. II.4 : *Sigma protocol* pour un langage $\{(x; w) \mid (x, w) \in \mathrm{R}\}$

Définition II – 6. Un protocole interactif (P, V) est un ***sigma protocol*** pour un langage binaire L s'il est en 3 tours de la forme donnée par la figure II.4 et satisfait les trois conditions suivantes :

- ***Perfect completeness*** : Si $x \in \mathrm{L}$, alors (P, V) accepte avec probabilité 1 ;
- ***Special soundness*** : Pour tout énoncé x, à partir de toute paire de conversations acceptées (t, c, z), (t, c', z') avec $c \neq c'$, on peut calculer en temps polynomial un témoin w d'appartenance de x à L ;
- ***Special honest-verifier zero-knowledge*** (SHVZK) : Il existe un algorithme polynomial qui sous l'entrée x et un défi c sort (t, c, z) distribué comme une conversation acceptée entre P et V honnêtes sous l'entrée x.

Théorème II – 7. Un *sigma protocol* avec espace de challenge $\mathscr{C}$ est une preuve interactive de connaissance HVZK pour le langage L avec erreur de *soundness* $1/|\mathscr{C}|$.

Démonstration. Il est clair que l'on a les propriétés de *completeness* et d'HVZK en prenant $c \xleftarrow{\$} \mathscr{C}$.

De même, le protocole est *sound* avec erreur $1/|\mathscr{C}|$. En effet, soit P^* tout puissant tel que $(\mathrm{P}^*, \mathrm{V})$ accepte avec probabilité strictement supérieure à $1/|\mathscr{C}|$ un $x \notin \mathrm{L}$. Si pour chaque choix d'engagement t par P^* il n'y avait qu'un seul défi c possible qui conduise V à accepter, alors la probabilité de succès de P^* ne serait que de $1/|\mathscr{C}|$: la probabilité que V choisisse ce bon défi c. Il existe donc un engagement t avec au moins deux défis c, c' distincts tel que V accepte les 2 conversations. Par la propriété de *special soundness* cela veut dire que $x \in \mathrm{L}$ et on a une contradiction. On a donc une preuve interactive pour L, HVZK avec *erreur de soundness* $1/|\mathscr{C}|$.

On peut en fait montrer que l'on a une preuve de connaissance de manière similaire (mais bien plus technique) à la preuve du protocole QR. C'est-à-dire que si la probabilité que $(\mathrm{P}^*, \mathrm{V})$ accepte un énoncé x est strictement supérieure à $1/|\mathscr{C}|$, alors on peut définir un extracteur similaire, qui rembobine P^* jusqu'à trouver – en temps moyen polynomial – un engagement t, et des défis c et c' avec $c \neq c'$, tel que les conversations (t, c, z), et (t, c', z') soient acceptées. La *special soundness* permet de conclure sur l'extraction. □

Un *sigma protocol* n'a pas la propriété de *zero-knowledge* (c'est-à-dire vis-à-vis d'un vérificateur V^* malveillant). De plus, pour avoir une erreur de *soundness* négligeable, l'espace des défis $\mathscr{C}$ doit être de taille exponentielle, ce qui rend impossible pour un simulateur de deviner le défi choisi par V^* comme on l'a vu pour le protocole de Schnorr. Cependant, la propriété SHVZK est suffisante pour de nombreuses applications, comme on le verra ensuite. D'autre part, il est possible avec un léger surcoût de transformer un *sigma protocol* en preuve interactive *zero-knowledge*. Pour cela, on utilise un *commitment* inconditionnellement *hiding* avec un espace de message correspondant à l'espace des défis $\mathscr{C}$. Pour empêcher que V^* choisisse le défi en fonction de l'engagement t du prouveur, on demande au vérificateur de mettre en gage son défi en début de protocole.

Entrée : x

- $P \leftarrow V$: V choisit $c \xleftarrow{\$} \mathscr{C}$ et envoie à P une mise en gage C de c
- $P \rightarrow V$: P envoie t à V
- $P \leftarrow V$: V envoie à P une ouverture de la mise en gage
- $P \rightarrow V$: P vérifie l'ouverture et s'arrête si elle est invalide. Sinon il envoie z à V.

Sortie : V renvoie accept ou reject à partir de (x, t, c, z)

Fig. II.5 : ZKPoK à partir d'un *sigma protocol* pour un langage $\{(x; w) \mid (x, w) \in R\}$

Théorème II – 8. Si le *commitment* est inconditionnellement *hiding* et le protocole sous-jacent est un *sigma protocol*, alors le protocole de la figure II.5 est une preuve interactive *zero-knowledge* avec erreur de soundness $1/|\mathscr{C}|$.

Démonstration. Les propriétés de *completeness* et de *soundness* sont clairement héritées du *sigma protocol* car le *commitment* est inconditionnellement *hiding* donc P^* tout puissant n'a pas plus d'information.

Pour la propriété de *zero-knowledge*, on définit le simulateur S suivant. Le simulateur invoque tout d'abord V^* pour obtenir la mise en gage C du défi. Ensuite, S utilise le simulateur $\tilde{S}$ SHVZK du *sigma protocol* à partir d'un défi c quelconque afin d'obtenir un premier message t_0 que S donne à V^* obtenant ainsi l'ouverture o de C. Si elle est invalide, S s'arrête. Sinon, S réutilise $\tilde{S}$ avec le c contenu dans l'ouverture pour obtenir (t, c, z). Le simulateur S invoque de nouveau V^* avec ce t, si l'ouverture de la mise en gage donne $c' \neq c$, S échoue, sinon il retourne la conversation : (C, t, o, z).

Idées de la preuve : Comme $\tilde{S}$ est un simulateur SHVZK, t_0 et t suivent la même distribution donc V^* envoie une ouverture avec la même distribution dans les deux cas (celle correspondant à une conversation de P avec V^*). De plus, la propriété de *binding* (calculatoire face à V^* polynomial) conduit à $c = c'$ sauf avec probabilité négligeable donc S n'échoue qu'avec probabilité négligeable. Ainsi le simulateur produit une conversation indistinguable de l'originale en temps polynomial. □

Ce protocole ne permet pas d'avoir une preuve de connaissance. Comme V met en gage c, on ne peut pas rembobiner P^* après qu'il a envoyé t pour lui donner un $c' \neq c$. Cependant, on peut utiliser un *commitment* à trappe : pour les utilisateurs normaux, il se comporte comme un commitment classique, mais pour celui qui connaît la trappe, il est possible d'ouvrir une mise en gage sur n'importe quelle autre valeur). Ceci permet d'utiliser l'extracteur du *sigma protocol* et la propriété de *special soundness* pour extraire le témoin. On obtient le même protocole, où le prouveur a la trappe du *commitment* et l'envoie à V dans la dernière étape avec z. Le vérificateur V vérifie de plus que la trappe est valide. Ainsi l'extracteur en disposera dès la première conversation acceptée, et dans la recherche d'une conversation acceptée avec le même engagement il pourra utiliser la trappe pour ouvrir sur n'importe quel autre défi.

Le *commitment* de Pedersen permet de définir un *commitment* à trappe, la trappe étant le logarithme discret de h en base g. Ainsi on peut se contenter d'un *sigma protocol* qui n'est « que » HVZK. Pour un léger surcoût, on peut le transformer en protocole *zero-knowledge*.

4. Preuves non interactives et signatures

Les preuves que l'on a vues nécessitent une interaction entre prouveur et vérifieur. Dans beaucoup de cas d'utilisation, cette interaction n'est pas possible ou le prouveur souhaite convaincre plusieurs personnes qu'un énoncé relatif à une donnée qu'il a créée est valide (un chiffré par exemple) sans devoir refaire le protocole avec chaque vérifieur. Les preuves non interactives vont permettre cela. L'idée est de transformer un protocole interactif en remplaçant les défis envoyés par le vérifieur par des hachés. Pour montrer qu'une telle approche fonctionne, on utilise le modèle de l'oracle aléatoire.

Modèle de l'oracle aléatoire

Le modèle de l'oracle aléatoire (*Random Oracle Model*, ROM) a été proposé par Bellare et Rogaway en 1993 pour idéaliser les fonctions de hachages. Une fonction de hachage h utilisée dans un protocole cryptographique est modélisée comme suit : chaque fois que l'on veut le haché de r, c'est-à-dire la valeur $h(r) \in \mathscr{H}$, l'espace des hachés, on soumet r à un oracle modélisant h qui répond par une valeur de $\mathscr{H}$ obtenue avec probabilité uniforme indépendamment des valeurs précédemment demandées. Par contre, si on a déjà soumis r à l'oracle pour un certain r, alors on obtiendra la même valeur $h(r)$.

Si h est une fonction de $\{0,1\}^k$ à valeurs dans $\{0,1\}^n$, on peut aussi de manière équivalente la prendre avec probabilité uniforme parmi toutes les fonctions possibles en début de protocole. On peut aussi simuler parfaitement l'oracle avec un dictionnaire :

$h(r)$:
Si $\mathrm{List}_h[r]$ n'est pas définie :
  $\mathrm{List}_h[r] \xleftarrow{\$} \{0,1\}^n$
Retourner $\mathrm{List}_h[r]$

Un acteur d'un protocole dans le modèle de l'oracle aléatoire n'a donc aucune information sur $h(r)$ (toutes les valeurs sont possibles avec probabilité uniforme), sauf si elle a demandé cette valeur à l'oracle aléatoire.

En pratique, étant donné une preuve d'un protocole dans le modèle de l'oracle aléatoire, on remplace h par une fonction de hachage cryptographique (SHA2, SHA3) et on obtient donc une sécurité heuristique.

NIZK

On définit une **preuve non interactive** de manière similaire à une preuve interactive, mais avec maintenant des algorithmes probabilistes polynomiaux P et V classiques. La propriété de *completeness* s'adapte immédiatement. Pour la *soundness*, un adversaire polynomial ne doit pouvoir trouver qu'avec probabilité négligeable un (x, π) avec $x \notin \mathrm{L}$ et π une preuve acceptée. Pour l'aspect *zero-knowledge*, on doit donner plus de pouvoir au simulateur. On définit donc la propriété de NIZK dans le modèle de l'oracle aléatoire, par l'existence d'un simulateur S qui simule à la fois des preuves π et les réponses à l'oracle aléatoire. De plus, un distingueur ne doit pouvoir en temps polynomial distinguer les simulations de S de véritables générations de preuves et de véritables réponses d'un oracle aléatoire.

Voyons le cas des *sigma protocol*. Avec les notations de la figure II.4, on introduit un oracle aléatoire $h : \mathrm{X} \times \mathrm{T} \mapsto \mathscr{C}$ où X désigne l'ensemble des énoncés et T l'ensemble des engagements t. On remplace le défi $c \xleftarrow{\$} \mathscr{C}$ du vérificateur honnête par $c := h(x, t)$, une requête à l'oracle aléatoire. La génération de la preuve non interactive de (x, w) devient donc $\pi := (t, z)$. C'est la transformation (heuristique) de **Fiat-Shamir** (1986), formalisée ensuite par Bellare et Rogaway (1993) avec le modèle de l'oracle aléatoire.

Plus tard, un vérificateur peut vérifier π de manière non interactive en appelant l'oracle aléatoire pour obtenir $c = h(x, t)$ et en vérifiant (t, c, z) comme dans le *sigma protocol*.

Remarquons que l'on peut souvent produire des preuves plus compactes en définissant $\pi := (c, z)$. La vérification consiste alors à d'abord reconstruire t à partir de l'équation de vérification du *sigma protocol* et de x, c, z (c'est souvent possible, comme dans le protocole de Schnorr où $t = g^z h^{-c}$ avec $x = (g, h)$). Une fois t reconstruit, on valide la preuve si et seulement si $c = h(x, t)$.

Il est clair que la transformation de Fiat-Shamir donne la propriété de *completeness*. Pour la *soundness*, l'idée est que dans le modèle aléatoire un prouveur tricheur n'a pas plus d'avantage que dans le *sigma protocol* : le haché de (x, t) est indéterminé pour lui, tant qu'il ne l'a pas demandé à l'oracle. Il pourrait cependant demander le haché pour plusieurs choix de t afin d'obtenir un challenge qui l'arrange, mais si $|\mathscr{C}|$ est exponentiellement grand et que le prouveur est polynomial, on montre que cela ne lui procure pas d'avantage.

De même, on montre que cette construction est bien *zero-knowledge*. Le simulateur S fonctionne ainsi : la simulation de l'oracle aléatoire sur les requêtes (x, t) suit la simulation classique vue plus haut. Pour simuler une preuve pour un énoncé x^*, S tire un défi c^* uniforme et utilise le simulateur SHVZK du *sigma protocol*, qui produit (t^*, z^*). Il faut ensuite gérer le fait que ce c^* doit être la réponse de la simulation de l'oracle aléatoire pour la requête (x^*, t^*) et qu'il pourrait y avoir une incohérence si cette requête a déjà été effectué. On montre que cela arrive avec probabilité négligeable si dans le *sigma protocol* les engagements sont tirés uniformément dans un espace T de taille exponentielle.

Signature numérique

On définit un schéma de signature comme la donnée de trois algorithmes : une génération de clef qui produit un couple clef de vérification (publique) et clef de signature (privée) ; un algorithme de signature d'une chaîne de bits m avec la clef de signature ; un algorithme de vérification, prenant en entrée un message m, une clef de vérification et une signature et retournant accept ou reject.

La méthode de Fiat-Shamir permet de construire des schémas de signatures à partir d'un *sigma protocol*. On suppose que $(x, w) \in \mathrm{R}$ est une relation difficile, c'est-à-dire qu'il est difficile de retrouver w à partir de x (par exemple $x = g^w$ comme dans le schéma de Schnorr). La clef publique est x, la clef privée w. Pour signer m, on utilise le prouveur du *sigma protocol* avec $c := h(m, t)$ avec h un oracle aléatoire. La signature de m est $\sigma := (t, z)$ (ou (c, z)). Pour la vérifier, on utilise la vérification du *sigma protocol*.

On montre que la signature obtenue est EUF-CMA (*existentially unforgeable under a chosen message attack*) dans le modèle de l'oracle aléatoire sous l'hypothèse que R est une relation difficile : étant donnée une clef publique, l'adversaire peut demander les signatures des messages de son choix à un oracle de signature et il ne doit pouvoir émettre un couple valide (m, σ) avec m non demandé à l'oracle de signature qu'avec probabilité négligeable.

L'idée de la preuve est de se servir d'un tel attaquant pour calculer w à partir de x. Pour répondre aux requêtes de signature sans connaître w on se sert du simulateur SHVZK du *sigma protocol* en « programmant » l'oracle aléatoire comme vu au dessus. Ensuite, si l'attaquant sait produire une signature, il doit le faire en cassant la *knowledge soundness* du *sigma protocol*, on va donc pouvoir utiliser l'extracteur pour retrouver w en faisant un pari que les requêtes à l'oracle aléatoire comme plus haut.

On obtient les signatures Schnorr, avec cette transformation sur le protocole de... Schnorr ! La clef privée est w et la clef publique $h = g^w$. Pour signer $m \in \{0,1\}$, Alice choisit aléatoirement de manière uniforme $r \in \mathbf{Z}/q\mathbf{Z}$, calcule $t = g^r$, et pose $c = \mathrm{H}(m, t)$, et calcule $z = r + wc$ dans $\mathbf{Z}/q\mathbf{Z}$. La signature est (c, z).

La vérification consiste à recalculer t en utilisant l'équation $t = g^z h^{-c}$, puis à vérifier $c : c \stackrel{?}{=} \mathrm{H}(m, t)$. Ces signatures sont EUF-CMA sous l'hypothèse DL.

Chapitre III

Chiffrement linéairement homomorphe et à seuil

1. Chiffrement linéairement homomorphe

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

- EvalAdd est un algorithme 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 de $m + m'$;
- EvalScal est un algorithme 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 de $a.m := m + \cdots + m$ (a fois).

Remarques. On veut parfois que les chiffrés obtenus soient aléatoires avec la même distribution que si l'on avait appliqué l'algorithme de chiffrement sur le message clair (connaissant les chiffrés d'origine). Les deux algorithmes EvalAdd et EvalScal sont alors probabilistes. Sur les exemples que l'on va voir cela revient à ajouter un chiffré de 0.

Notons que des schémas linéairement homomorphes sont malléables et ne peuvent pas être IND – CCA2. On se contente en général d'une sécurité IND – CPA.

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 : $c = (-1)^m r^2$, $c' = (-1)^{m'} r'^2$, avec $c, c' \in (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$. L'algorithme EvalAdd consiste à calculer $c'' := cc'$. En effet, $c'' = (-1)^{m+m'}(rr')^2$ est un chiffré de $m + m' \in \mathbf{Z}/2\mathbf{Z}$: si $m = m'$, c'' est un carré 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. Notant qu'en prenant $r'' \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$, $c''r''^2$ sera un chiffré aléatoire de $m + m'$.

2. Le schéma de Paillier vu en TD : $\mathscr{M} = (\mathbf{Z}/\mathrm{N}\mathbf{Z}, +)$. Soit $m, m' \in \mathscr{M}$, et leurs chiffrés : $c = (1+\mathrm{N})^m r^\mathrm{N}$, $c' = (1+\mathrm{N})^{m'} r'^\mathrm{N}$, avec $c, c' \in (\mathbf{Z}/\mathrm{N}^2\mathbf{Z})^\times$. L'algorithme EvalAdd consiste à calculer $c'' := cc'$. En effet, $c'' = (1+\mathrm{N})^{m+m'}(r'r')^\mathrm{N}$ est bien un chiffré aléatoire de $m + m' \in \mathbf{Z}/\mathrm{N}\mathbf{Z}$ car $(1 + \mathrm{N})$ est d'ordre N modulo N^2.

   Pour l'algorithme EvalScal, étant donné c et a, on ressort $c^a = (1+\mathrm{N})^{ma}(r^a)^n$. On peut de même multiplier les chiffrés obtenus par $(r'')^\mathrm{N}$ avec $r'' \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}^2\mathbf{Z})^\times$ pour obtenir un chiffré aléatoire.

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 (liée à l'addition de $\mathbf{Z}$), en chiffrant m par $(c_1, c_2) =$

$(g^y, g^m h^y)$. C'est la variante « Elgamal dans l'exposant ». Mais dans ce cas, on doit se limiter à de petits m (et 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 (***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 (TFHE, CKKS) est basée sur des problèmes algorithmiques utilisant les réseaux euclidiens (dérivés du problème *Learning With Errors*, LWE, de Regev).

2. Une application : le vote électronique

Supposons que ℓ électeurs veuillent voter à un référendum. On va utiliser un chiffrement linéairement homomorphe. Une autorité a un couple (pk, sk).

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.

En déchiffrant c avec sk, l'autorité retrouvera $\sum_{i=1}^{\ell} m_i \in \mathscr{M}$ de laquelle elle doit pouvoir déduire $\sum_{i=1}^{\ell} m_i$ dans $\mathbf{Z}$. Ainsi elle 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.

Avec le chiffrement de Paillier, l'espace des clairs est $\mathbf{Z}/\mathrm{N}\mathbf{Z}$, on suppose donc que $\ell < \mathrm{N}$ pour pouvoir retrouver la somme dans $\mathbf{Z}$. Ce n'est pas une restriction en pratique (N étant un module RSA il fait au moins 2048 bits, soit plus de 600 chiffres décimaux). On peut utiliser aussi Elgamal dans l'exposant (solution utilisée dans le système de vote Belenios), en effet, un logarithme discret g^m avec m de l'ordre de quelques millions se calcule en temps raisonnable même avec l'algorithme naïf.

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 (typiquement en non interactif).
- 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. Avec Elgamal dans l'exposant, on peut tout simplement voter 0 ou 1 pour chaque candidat et avoir k chiffrés.

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

Définitions

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, $s_1, \dots, s_n$, et envoie la part s_i pour $i = 1, \dots, n$ sur un canal sécurisé au participant i.
- Un algorithme de reconstruction : certains participants rendent publiques leur part et le secret s est reconstruit en temps polynomial 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 $\mathbf{F}_2^k$ 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 $\mathbf{F}_2^k$ indépendantes du secret s. Ainsi, $n-1$ parts ne donnent aucune information sur le secret.

En pratique, il est souvent plus flexible de pouvoir reconstruire le secret avec moins de n parts. On obtient alors un **partage de secret à seuil** de paramètre $[t, n]$ avec $t \leqslant n$ (t pour *threshold*) : avec $t-1$ parts parmi n on n'obtient aucune information sur le secret, et t parts permettent de reconstruire le secret. Le partage du paragraphe précédent a pour paramètre $[n, n]$. 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.

Partage de secret de Shamir (1979)

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 suppose $q > n$ et on considère fixés (et publics) n points distincts de $\mathbf{F}_q^*$, $x_1, x_2, \dots, x_n$. 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 $g(X) = \sum_{j=1}^{t} s_j \ell_j(X)$, 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 une correspondance entre codes linéaires et partages de secret linéaires, les codes MDS correspondant aux partages de secret à seuil.

Partage de secret vérifiable

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. Lors du partage, le distributeur publie $A_0 := g^s := g^{a_0}$, $A_1 := g^{a_1}, \dots, A_{t-1} := g^{a_{t-1}}$. 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

Un **chiffrement à seuil** est un chiffrement asymétrique classique où les procédures de génération de clefs et de déchiffrement suivent la logique du partage de secret avec n participants.

La génération de clef est faite par un distributeur de confiance qui envoie des parts de la clef privée aux n participants et produit la clef publique. On peut aussi vouloir rendre cette partie totalement distribuée en éliminant ce tiers de confiance : les participants exécutent un protocole multipartite qui leur permet de construire leur part sans information sur les parts des autres participants (et sur la clef secrète). On parle de *distributed key generation* (DKG).

Le déchiffrement est aussi partagé. On utilise un seuil $t \leqslant n$: t participants parmi n peuvent déchiffrer un chiffré donné (et pas plus, en particulier ils ne reconstituent pas la clef secrète) ; et $t-1$ ne peuvent rien faire et n'apprennent rien sur le déchiffrement.

Elgamal à seuil, Pedersen (1991)

On combine le chiffrement Elgamal avec le partage de secret vérifiable de Feldman.

Pour un paramètre de sécurité k, on considère un groupe cyclique $\mathbf{G}$ son ordre q premier avec $|q| = k$ et un générateur g.

L'algorithme distribué KeyGen choisit x uniforme dans $\mathbf{Z}/q\mathbf{Z}$ et calcule et publie $h = g^x$ qui sera la clef publique. Il partage x par le schéma de Feldman. En particulier, chaque participant reçoit une part s_i. On publie également $h_i := g^{s_i}$.

L'algorithme de chiffrement est inchangé : sur l'entrée (pk, m), on choisit r aléatoirement dans $\mathbf{Z}/q\mathbf{Z}$ et retourne $c = (g^r, mh^r)$ pour un $m \in \mathrm{G}$.

Pour le déchiffrement de (c_1, c_2), t participants parmi les n collaborent. Le participant i calcule $w_i := c_1^{s_i}$ et l'envoie aux autres participants. Il prouve également que son résultat est correct en prouvant en *zero-knowledge* (en général en non interactif) que $\log_{c_1} w_i = \log_g h_i$.

Pour retrouver le résultat, il faut pouvoir calculer $c_2(c_1^x)^{-1}$. Les t participants vont construire c_1^x en utilisant la reconstruction du partage de Shamir « dans l'exposant ». Pour rappel, x peut s'écrire comme une combinaison linéaire $x = \sum_{i \in \mathscr{I}} \lambda_i s_i$ pour tout sous ensemble $\mathscr{I}$ de t participants. Les λ_i, publics, correspondent aux polynômes d'interpolation de Lagrange évalués en 0.

À partir des w_i, les participants peuvent donc tous calculer

$$\prod_{i \in \mathscr{I}} w_i^{\lambda_i} = \prod_{i \in \mathscr{I}} c_1^{\lambda_i s_i} = c_1^{\sum_{i \in \mathscr{I}} \lambda_i s_i} = c_1^x.$$

Retour sur le vote

Un schéma de chiffrement linéairement homomorphe IND – CPA à seuil résout le problème de confidentialité du schéma de vote esquissé en section 2. La clef secrète de déchiffrement est partagée entre plusieurs autorités. Au moins t autorités doivent collaborer pour pouvoir déchiffrer. Si le nombre d'autorités malveillantes est strictement inférieur à t, alors seul le chiffré c correspondant à l'évaluation homomorphe de la somme des votes est déchiffré, et les chiffrés des votes individuels ne le sont pas. De plus, les preuves *zero-knowledge* générées pendant le déchiffrement distribué assurent que le résultat du vote est correct.

Chapitre IV

Sécurité IND – CCA

1. Introduction

On s'intéresse toujours à la sécurité du chiffrement à clef publique. On considère maintenant le modèle CCA (*Chosen-Ciphertext Attacks*), attaques à chiffrés choisis, où l'attaquant peut obtenir les déchiffrements de chiffrés de son choix. Ce modèle d'attaque est motivé par différents scénarios.

Supposons qu'Alice envoie un message chiffré, c^*, à Bob. Un attaquant observe c^* et veut connaître le message clair correspondant m^*. On considère maintenant un attaquant actif (contrairement à l'attaquant IND – CPA qui ne faisait qu'écouter la ligne). Il peut créer d'autres chiffrés c (fonction ou non de c^*) et les envoyer à Bob. Suivant la réaction de Bob, qui déchiffre le chiffré c grâce à sa clef privée pour obtenir un message m, l'attaquant pourrait obtenir de l'information sur m (si ce n'est m tout entier si Bob lui répond en clair en citant m). Il peut tenter de s'aider de ces informations pour décrypter ensuite c^* ou obtenir de l'information sur m^*. Bob agit comme un oracle de déchiffrement pour l'attaquant.

Une autre motivation pour ce modèle d'attaque est l'attaque de Bleichenbacher (1998) sur les protocoles basés sur le standard de chiffrement RSA PKCS #1 v1.5. Ce chiffrement utilise une variante probabiliste de RSA avec un remplissage : $(m||0000\,0000||r||0100\,0000||0000\,0000)^e \mod N$, où r représente un aléa et où le message, une chaîne de bits, est codé dans les bits de poids forts. Si lors du déchiffrement l'application de la fonction trappe RSA ne donne pas un entier de la bonne forme, un message d'erreur est retourné (c'est le cas lors de l'établissement d'une connexion dans SSL V.3.0).

Bleichenbacher a démontré comment utiliser ce bit d'information pour monter une attaque. Étant donné un chiffré c^* de m^* à attaquer, il produit des chiffrés c reliés à c^* et en observant la réponse d'un serveur de déchiffrement il montre comment retrouver m^*. Ici on obtient d'un oracle une information partielle (un bit) lors du déchiffrement. Si on sait se protéger d'attaques dans lesquelles un oracle de déchiffrement donne la totalité du message clair, alors on sera protégé de ces attaques à la Bleichenbacher.

Cette attaque a donc motivé le modèle CCA et l'introduction d'un standard RSA prouvé sûr dans ce modèle : RSA-OAEP (*Optimal Asymmetric Encryption Padding*, transformation introduite par Bellare et Rogaway en 1994), prouvé sûr par Fujisaki, Okamoto, Pointcheval, Stern (2001), standardisé dans PKCS #1 v2.1, en 2002.

2. Définition

2.1. IND – CCA2

Définition IV – 1 (IND – CCA2, Rackoff et Simon, 1991)**.** 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 note $\mathscr{O}_{\mathsf{Dec}}$ un oracle de déchiffrement, c'est à dire l'algorithme $\mathsf{Decrypt}(sk, \cdot)$. On définit l'expérience d'indistinguabilité CCA2, $\mathbf{Exp}^{\mathsf{IND-CCA2}}_{\Pi,k}(\mathscr{A})$:

1. On lance l'algorithme $\mathsf{KeyGen}(1^k)$ pour obtenir les clefs (pk, sk)

2. $\mathscr{A}_1$ reçoit pk et peut interagir avec l'oracle $\mathscr{O}_{\mathsf{Dec}}$ en lui soumettant des chiffrés et en recevant les déchiffrements correspondant. $\mathscr{A}_1$ 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. $\mathscr{A}_2$ peut interagir avec l'oracle $\mathscr{O}_{\mathsf{Dec}}$ avec la restriction qu'il ne peut soumettre $c^\star$ à $\mathscr{O}_{\mathsf{Dec}}$. Il sort ensuite 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

$$\mathbf{Adv}^{\mathsf{IND-CCA2}}_{\Pi,k}(\mathscr{A}) = \left|\Pr\left(\mathbf{Exp}^{\mathsf{IND-CCA2}}_{\Pi,k}(\mathscr{A}) = 1\right) - \frac{1}{2}\right|.$$

Le schéma Π est sûr au sens IND – CCA2 si pour tout algorithme polynomial probabiliste $\mathscr{A}$ il existe une fonction négligeable ν telle que pour k assez grand,

$$\mathbf{Adv}^{\mathsf{IND-CCA2}}_{\Pi,k}(\mathscr{A}) \leqslant \nu(k).$$

On peut définir aussi une sécurité IND – CCA1 (Naor et Yung, 1990) (attaque non adaptative au lieu d'attaque adaptative), $\mathscr{A}$ a alors accès à l'oracle $\mathscr{O}_{\mathsf{Dec}}$ uniquement à l'étape 2. On parle aussi de ***lunchtime attack*** : durant la pause déjeuner, Oscar subtilise l'appareil stockant la clef privée de Bob et lui soumet des requêtes de déchiffrement pour gagner de l'information en vue de faire une attaque. Il rend ensuite l'appareil à Bob sans se faire remarquer et essaye de se servir de l'information acquise pour attaquer un chiffré c^* envoyé à Bob.

Comme pour les attaques CPA, on peut montrer que la sécurité sémantique dans le cadre d'attaque CCA2 est équivalente à la sécurité IND – CCA2. Par contre, contrairement au cadre CPA, la non malléabilité CCA2 est maintenant aussi équivalente à la sécurité IND – CCA2 (Bellare, Desai, Pointcheval, Rogaway 1998). Ceci donne une autre motivation pour IND – CCA2 qui correspond ainsi à la notion regroupant de nombreuses propriétés de sécurité pour le chiffrement. On peut également définir des notions de sécurité TB – CCA2 et OW – CCA2. Pour résumer, on a

$$\mathsf{TB-CCA2} \Leftarrow \mathsf{OW-CCA2} \Leftarrow \mathsf{IND-CCA2} \Leftrightarrow \mathsf{NM-CPA} \Leftrightarrow \mathsf{Sem-sec-CCA2}.$$

Idées de construction

Pour atteindre le niveau de sécurité IND–CCA2, l'idée généralement utilisée est de rajouter de l'intégrité aux chiffrés : on veut pouvoir s'assurer que les chiffrés ont été correctement formés en utilisant l'algorithme de chiffrement. Ainsi, un adversaire ne peut produire un chiffré c correct sans connaître le clair associé. Un oracle de déchiffrement ne lui apporte donc aucune information qu'il n'avait déjà.

Pour cela, on utilise en général des fonctions de hachage. Lors du chiffrement, un haché d'une quantité se déduisant de l'aléa et/ou du message est inclus dans le chiffré. Le message est déchiffré uniquement si le haché est correct.

À partir d'Elgamal, Cramer et Shoup ont proposé un schéma IND – CCA2 sûr sous l'hypothèse DDH en 1998, avec des hypothèses standard sur les fonctions de hachage utilisées (familles de fonctions de hachage universelles à sens unique) : c'est un schéma dit dans le modèle standard.

Nous allons voir des constructions plus efficaces mais en idéalisant les fonctions de hachage utilisées avec le modèle oracle aléatoire.

3. Constructions dans le ROM

3.1. Un RSA IND – CPA

C'est une construction introduite par Bellare et Rogaway (1993) Soit k un paramètre de sécurité et GenRSA qui prend en entrée 1^k et ressort les paramètres N, e, d de RSA. Soit h un oracle aléatoire pouvant prendre en entrée des éléments de $(\mathbf{Z}/N\mathbf{Z})^\times$ pour tout N et pouvant retourner des valeurs de $\{0,1\}^k$ pour tout k.

Pour chiffrer $m \in \{0,1\}k$, on choisit $r \xleftarrow{\$} (\mathbf{Z}/N\mathbf{Z})^\times$ et on calcule

$$c = (c_1, c_2) = (r^e, h(r) \oplus m).$$

Pour déchiffrer, on calcule $r = c_1^d$ dans $(\mathbf{Z}/N\mathbf{Z})^\times$ grâce à la clef secrète d puis $m = c_2 \oplus h(r)$.

Ce schéma et le théorème ci-dessous se généralise aisément en remplaçant RSA par une permutation à trappe quelconque (ce sera également le cas pour la version IND – CCA2 que l'on verra ensuite).

Théorème IV – 2. Le schéma ci-dessus est IND–CPA dans le modèle de l'oracle aléatoire sous l'hypothèse RSA.

Voyons tout d'abord une preuve intuitive. Le fait que RSA soit à sens unique, empêche d'obtenir r grâce à c_1. Cependant l'adversaire peut obtenir des bits d'information sur r (mais pas r tout entier). Quoi qu'il en soit, la valeur $h(r)$ restera totalement indéterminée pour l'adversaire car il ne peut soumettre r à l'oracle aléatoire h. Comme c_2 résulte d'un *one-time-pad* de $h(r)$ indéterminé avec m, l'adversaire n'obtient aucune information sur m et le schéma est sémantiquement sûr.

Démonstration. On fait une preuve par jeux. Soit $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ un adversaire IND – CPA.

Jeu 0 C'est le jeu IND – CPA :

1. KeyGen(1^k) donne les clefs $pk = (N, e)$, $sk = d$.
2. $\mathscr{A}_1$ reçoit pk et retourne (m_0, m_1, s) deux messages de $\{0,1\}^k$ 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 $m_{b^\star}$:

$$c^\star \leftarrow \mathsf{Encrypt}(pk, m_{b^\star}) = ({r^\star}^e, h(r^\star) \oplus m_{b^\star}), r^\star \xleftarrow{\$} (\mathbf{Z}/N\mathbf{Z})^\times$$

   et on donne $(s, c^\star)$ à $\mathscr{A}_2$
5. $\mathscr{A}_2$ sort un bit b

Pendant tout le jeu, $\mathscr{A}$ peut interroger l'oracle aléatoire h.

On note S_i l'évènement « $b = b^\star$ » dans ce jeu et ceux à venir de telle sorte que

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

Jeu 1 On laisse inchangé le jeu 0 mais on simule maintenant l'oracle aléatoire avec la simulation classique : à chaque fois que $\mathscr{A}$ fait une requête x, on lui répond en exécutant la fonction suivante :

```
h_sim(x) :
    Si List_h[x] n'est pas défini :
        List_h[x] ←$ (Z/NZ)^×
    Retourner List_h[x]
```

La simulation est parfaite, on a donc

$$\Pr(S_0) = \Pr(S_1).$$

Jeu 2 On génère $r^\star \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$ au début du jeu plutôt qu'au point 4. On y génère aussi $z^\star \xleftarrow{\$} \{0,1\}^k$ que l'on utilise pour le chiffré *challenge* : $c^\star = ({r^\star}^e, z^\star \oplus m_{b^\star})$. De plus, on rajoute une règle dans la simulation de l'oracle : si $\mathscr{A}$ fait une requête $x = r^\star$, on lui répond $z^\star$, c'est à dire qu'on pose $List_h[r^\star] := z^\star$.

1. $r^\star \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$; $z^\star \xleftarrow{\$} \{0,1\}^k$; $List_h[r^\star] = z^\star$. $\mathsf{KeyGen}(1^k)$ donne les clefs $pk = (\mathrm{N}, e)$, $sk = d$.
2. $\mathscr{A}_1$ reçoit pk et retourne (m_0, m_1, s) avec deux messages de $\{0,1\}^k$ 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 $m_{b^\star}$:
$$c^\star = ({r^\star}^e, z^\star \oplus m_{b^\star})$$
et on donne $(s, c^\star)$ à $\mathscr{A}_2$
5. $\mathscr{A}_2$ sort un bit b

Avec la même simulation h_{sim} pour h avec $List_h[r^\star] := z^\star$. On a toujours

$$\mathrm{P}(\mathrm{S}_1) = \mathrm{P}(\mathrm{S}_2)$$

car on a défini de manière cohérente le haché de $r^\star$ dans tout le jeu.

Jeu 3 Même jeu, mais pour la simulation de l'oracle, on revient comme au jeu 1, en supprimant la règle pour $r^\star$.

1. $r^\star \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$; $z^\star \xleftarrow{\$} \{0,1\}^k$. $\mathsf{KeyGen}(1^k)$ donne les clefs $pk = (\mathrm{N}, e)$, $sk = d$.
2. $\mathscr{A}_1$ reçoit pk et retourne (m_0, m_1, s) avec deux messages de $\{0,1\}^k$ 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 $m_{b^\star}$:
$$c^\star = ({r^\star}^e, z^\star \oplus m_{b^\star})$$
et on donne $(s, c^\star)$ à $\mathscr{A}_2$
5. $\mathscr{A}_2$ sort un bit b

Avec la simulation h_{sim} pour h.

Comme $z^\star$ n'est plus utilisé que pour masquer $m_{b^\star}$, $\mathscr{A}$ n'a aucune information dessus, donc $c^\star$ masque totalement $m_{b^\star}$, la réponse b de $\mathscr{A}_2$ est donc indépendante de $b^\star$ et

$$\Pr(\mathrm{S}_3) = 1/2.$$

On note Q_2 (resp. Q_3) l'évènement « $\mathscr{A}$ demande $r^\star$ à l'oracle aléatoire » au jeu 2 (resp. au jeu 3). Quand Q_2 ne se produit pas, le jeu 2 est identique au jeu 3 quand Q_3 ne se produit pas (si ces événements se produisent, il y aurait alors une incohérence entre la valeur $h_{\text{sim}}(r^\star)$ apprise par l'adversaire et celle utilisée dans $c^\star$). On a donc $\Pr(\mathrm{S}_2 \wedge \neg\mathrm{Q}_2) = \Pr(\mathrm{S}_3 \wedge \neg\mathrm{Q}_3)$, et de plus $\Pr(\mathrm{Q}_2) = \Pr(\mathrm{Q}_3)$ puisque l'adversaire joue le même jeu tant que cet évènement n'arrive pas. On a donc, comme vu en I.1,

$$|\Pr(\mathrm{S}_2) - \Pr(\mathrm{S}_3)| \leqslant \Pr(\mathrm{Q}_3).$$

Remarquons que la valeur de $r^\star$ n'est plus utilisée dans le jeu 3, à part pour définir $c_1^\star$, et donc que $r^\star$ est la valeur telle que $c_1^\star = {r^\star}^e$. On montre maintenant que $\Pr(\mathrm{Q}_3)$ correspond au succès d'un attaquant $\mathscr{B}$ contre le problème RSA. On définit $\mathscr{B}$ comme suit.

$\mathscr{B}(\mathrm{N}, e, y^\star)$:

$z^\star \xleftarrow{\$} \{0,1\}^k$. $pk := (\mathrm{N}, e)$
$(m_0, m_1, s) \leftarrow \mathscr{A}_1(pk)$

$b^\star \xleftarrow{\$} \{0,1\}$
$c^\star = (y^\star, z^\star \oplus m_{b^\star})$
$b \leftarrow \mathscr{A}_2(s, c^\star)$
$\mathscr{B}$ retourne échec

Durant l'exécution de $\mathscr{A}$, $\mathscr{B}$ répond aux requêtes à l'oracle aléatoire avec la fonction suivante.

$h_{\mathrm{sim}}(x)$:
Si $x^e = y^\star$:
$\mathscr{B}$ s'arrête et retourne x comme solution au problème RSA
Si $List_h[x]$ n'est pas défini :
$List_h[x] \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$
Retourner $List_h[x]$ à $\mathscr{A}$.

L'algorithme $\mathscr{B}$ joue le même rôle du challenger dans le jeu 3 (pk est toujours une clef publique RSA et $y^\star$ a bien la même distribution que ${r^\star}^e$ dans $c_1^\star$. D'autre part, $\mathbf{Succ}^{\mathrm{RSA}}(\mathscr{B}) = \Pr(\mathrm{Q}_3)$. Au final, on a

$$|\Pr(\mathrm{S}_2) - \Pr(\mathrm{S}_3)| = |\Pr(\mathrm{S}_0) - \Pr(\mathrm{S}_3)| = |\Pr\left(\mathbf{Exp}^{\mathsf{IND-CPA}}_{\Pi,k}(\mathscr{A}) = 1\right) - 1/2| = \mathbf{Adv}^{\mathsf{IND-CPA}}(\mathscr{A}) \leqslant \mathbf{Succ}^{\mathrm{RSA}}(\mathscr{B}).$$

Ce qui prouve que le schéma est IND – CPA sous l'hypothèse RSA.

□

3.2. Un RSA IND – CCA

Remarquons que la construction précédente n'est pas IND – CCA2. En effet, étant donné le chiffré *challenge* $c^\star = ({r^\star}^e, h(r^\star) \oplus m_{b^\star})$, $\mathscr{A}_2$ peut demander à l'oracle de déchiffrement de déchiffrer $(c_1^\star, z)$ avec $z \xleftarrow{\$} \{0,1\}^k$ qui sera bien différent de $c^\star$ sauf avec probabilité négligeable. Il obtient ainsi $z \oplus h({c_1^\star}^d) = z \oplus h(r^\star)$. L'attaquant peut donc en déduire $h(r^\star)$ et donc $m_{b^\star}$.

Soit k un paramètre de sécurité et GenRSA qui prend en entrée 1^k et ressort les paramètres N, e, d de RSA. Soit h un oracle aléatoire pouvant prendre en entrée des éléments de $(\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$ pour tout N et pouvant retourner des valeurs de $\{0,1\}^k$ pour tout k. On considère maintenant un deuxième oracle aléatoire $g : (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times \times \{0,1\}^k \to \{0,1\}^k$.

Pour chiffrer $m \in \{0,1\}^k$, on choisit $r \xleftarrow{\$} (\mathbf{Z}/n\mathbf{Z})^\times$ et on calcule maintenant

$$c = (c_1, c_2, c_3) = \left(r^e, h(r) \oplus m, g(r, c_2)\right).$$

Pour déchiffrer, on calcule $r = c_1^d$ grâce à la clef secrète d puis on teste si $c_3 = g(r, c_2)$ si non on retourne $\perp$ si oui on retourne $m = c_2 \oplus h(r)$.

L'attaque précédente n'est plus possible. L'attaquant $\mathscr{A}_2$ ne pourra plus modifier $c_2^\star$ en gardant le haché $c_3^\star$ correct : il a besoin de la valeur de r pour mettre à jour ce haché.

Théorème IV – 3. Le schéma ci-dessus est IND – CCA2 dans le modèle de l'oracle aléatoire sous l'hypothèse RSA.

Intuitivement, l'ajout de c_3 rend l'oracle de déchiffrement inutile : si l'adversaire construit un chiffré correct, c'est-à-dire tel que le déchiffrement ne donne pas $\perp$, alors la valeur $g(r, c_2)$ est correcte. Ceci signifie que l'attaquant a demandé (r, c_2) à l'oracle aléatoire et donc qu'il connaît r. Il sait donc déjà déchiffrer en calculant $c_2 \oplus h(r)$. L'oracle de déchiffrement est donc inutile pour l'attaquant, et le chiffrement est donc sûr car on a montré qu'il était IND – CPA.

Démonstration. Soit $\mathscr{A} = (\mathscr{A}_1, \mathscr{A}_2)$ un adversaire IND – CCA2.

Jeu 0 C'est le jeu IND – CCA2.

1. KeyGen(1^k) donne les clefs $pk = (\mathrm{N}, e)$, $sk = d$.
2. $\mathscr{A}_1$ reçoit pk et retourne (m_0, m_1, s) deux messages de $\{0,1\}^k$ 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 $m_{b^\star}$:

$$c^\star \leftarrow \mathsf{Encrypt}(pk, m_{b^\star}) = \left(r^{\star e}, h(r^\star) \oplus m_{b^\star}, g(r^\star, c_2^\star)\right), r^\star \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$$

   et on donne $(s, c^\star)$ à $\mathscr{A}_2$
5. $\mathscr{A}_2$ sort un bit b.

Pendant tout le jeu, $\mathscr{A}$ peut interroger les oracles aléatoires h et g. Il peut également interroger un oracle de déchiffrement (sans restriction pour $\mathscr{A}_1$, pour des chiffrés différent de $c^\star$ pour $\mathscr{A}_2$).

On note S_i l'évènement « $b = b^\star$ » dans ce jeu et ceux à venir de telle sorte que

$$\Pr(\mathrm{S}_0) = \Pr\left(\mathbf{Exp}_{\Pi,k}^{\mathsf{IND-CCA2}}(\mathscr{A}) = 1\right).$$

Jeu 1 On laisse inchangé le jeu 0 mais on simule maintenant les oracles aléatoires avec la simulation classique.

$h_{\mathrm{sim}}(x)$:
  Si $\mathrm{L}ist_h[x]$ n'est pas défini :
    $\mathrm{L}ist_h[x] \xleftarrow{\$} (\mathbf{Z}/\mathrm{N}\mathbf{Z})^\times$
  Retourner $\mathrm{L}ist_h[x]$

$g_{\mathrm{sim}}(y, z)$:
  Si $\mathrm{L}ist_g[y, z]$ n'est pas défini :
    $\mathrm{L}ist_g[y, z] \xleftarrow{\$} \{0,1\}^k$
  Retourner $\mathrm{L}ist_g[y, z]$

La simulation est parfaite, on a donc

$$\Pr(\mathrm{S}_0) = \Pr(\mathrm{S}_1).$$

Jeu 2 On simule l'oracle de déchiffrement comme suit.

$\mathrm{D}_{\mathrm{sim}}(c_1, c_2, c_3)$:
  S'il existe, dans la liste des requêtes à g_{sim}, (r, c_2) tel que $\mathrm{L}ist_g[r, c_2] = c_3$ et $r^e = c_1$:
    Retourner $c_2 \oplus h_{\mathrm{sim}}(r)$,
  Sinon
    Retourner $\perp$

On ne déchiffre donc que des chiffrés pour lesquels $\mathscr{A}$ a demandé (r, c_2) à g_{sim} tel que $c_1 = r^e$ afin de poser $c_3 = g_{\mathrm{sim}}(r, c_2)$. De tels chiffrés sont corrects et sont déchiffrés de manière identique dans les jeux 1 et 2.

Examinons les cas où cette simulation n'est pas correcte. Supposons que lors de sa première requête de déchiffrement, seulement, $\mathscr{A}$ demande le déchiffrement d'un (c_1, c_2, c_3) sans avoir demandé (r, c_2) à g_{sim} où $r = c_1^d$. L'algorithme $\mathrm{D}_{\mathrm{sim}}$ retourne donc incorrect dans le jeu 2. Cependant dans le jeu 1, l'oracle de déchiffrement aurait interrogé g_{sim} sur (r, c_2). Cette valeur n'est pas fixée ($\mathscr{A}$ ne l'a pas demandée, l'oracle de déchiffrement n'a pas encore été interrogé donc n'a pas fixé de valeur, et remarquons que le haché de $(r^\star, c_2^\star)$ est fixé par $c^\star$ pour $\mathscr{A}_2$ mais c demandé par $\mathscr{A}_2$ est différent de $c^\star$ donc $(r, c_2) \neq (r^\star, c_2^\star)$). Avec probabilité $1/2^k$, g_{sim} aurait renvoyé c_3 dans le jeu 1 et le chiffré aurait été déchiffré. Il y donc une incohérence mais celle si n'arrive que si cet évènement de probabilité $1/2^k$ se produit dans les deux jeux. En procédant de même pour toutes les q requêtes de déchiffrement, on obtient, en utilisant I.1,

$$|\Pr(S_1) - \Pr(S_2)| \leqslant q/2^k.$$

Notons que $q/2^k$ est négligeable car le nombre de requêtes de $\mathscr{A}$ polynomial est lui aussi polynomial.

Jeu 3 On procède maintenant comme dans le preuve IND – CPA. On prend $r^\star \xleftarrow{\$} (\mathbf{Z}/N\mathbf{Z})^\times$ en début de jeu, $z^\star \xleftarrow{\$} \{0,1\}^k, c_3^\star \xleftarrow{\$} \{0,1\}^k$ et on pose le chiffré challenge $c^\star = (r^{\star e}, z^\star \oplus m_{b^\star}, c_3^\star)$.

1. $r^\star \xleftarrow{\$} (\mathbf{Z}/N\mathbf{Z})^\times$; $z^\star \xleftarrow{\$} \{0,1\}^k$; $c_3^\star \xleftarrow{\$} \{0,1\}^k$; KeyGen(1^k) donne les clefs $pk = (N, e)$, $sk = d$.
2. $\mathscr{A}_1$ reçoit pk et retourne (m_0, m_1, s) avec deux messages de $\{0,1\}^k$ 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 $m_{b^\star}$:
$$c^\star = (r^{\star e}, z^\star \oplus m_{b^\star}, c_3^\star)$$
et on donne $(s, c^\star)$ à $\mathscr{A}_2$
5. $\mathscr{A}_2$ sort un bit b

Comme dans la preuve IND – CPA on a maintenant

$$\Pr(S_3) = 1/2.$$

On note Q l'évènement « $\mathscr{A}$ demande $r^\star$ (tel que $c_1 = r^{\star e}$) à h ou un $(r^\star, c_2)$ à g ». Les jeux 2 et 3 procèdent de manière identique si Q n'arrive pas (sinon il y une incohérence possible comme vu dans la preuve IND – CPA. On a donc

$$|\Pr(S_2) - \Pr(S_3)| \leqslant \Pr(Q).$$

Comme dans le preuve IND – CPA on définit un attaquant $\mathscr{B}$ contre le problème RSA en jouant le rôle du challenger dans le jeu 3, en posant $c_1^\star := y^\star$, le défi RSA ,et en filtrant les requêtes aux oracles aléatoires pour trouver x tel que $x^e = y^\star$. On a ainsi $\mathbf{Succ}^{\mathrm{RSA}}(\mathscr{B}) = \Pr(Q)$.

Au final

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{IND-CCA2}}(\mathscr{A}) &= \left|\Pr\left(\mathbf{Exp}^{\mathsf{IND-CCA2}}(\mathscr{A}) = 1\right) - 1/2\right| \\
&= |\Pr(S_0) - \Pr(S_3)| \\
&\leqslant |\Pr(S_0) - \Pr(S_1)| + |\Pr(S_1) - \Pr(S_2)| + |\Pr(S_2) - \Pr(S_3)| \\
&\leqslant q_d/2^k + \mathbf{Succ}^{\mathrm{RSA}}(\mathscr{B}).
\end{aligned}$$

Ce qui prouve que le schéma est IND – CCA2 sous l'hypothèse RSA.

□

Le schéma de chiffrement obtenu a des chiffrés relativement long : pour un paramètre de sécurité $k = 128$, le modulo RSA N doit être de 3072 bits. Un message clair de 128 bits est donc chiffré en c de $3072 + 2 \times 128 = 3328$ bits. À ce niveau de sécurité, un schéma reposant sous l'hypothèse RSA doit avoir des chiffrés d'au moins 3072 bits. On aimerait ne pas avoir des chiffrés plus long. C'est ce que permet RSA-OAEP, pour *Optimal Asymmetric Encryption Padding*.

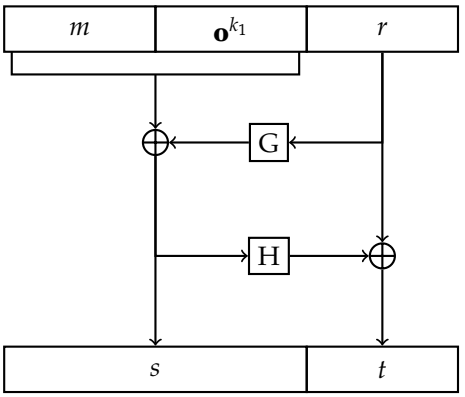
RSA-OAEP

La transformation OAEP consiste à appliquer deux tours de schémas de Feistel avec des oracles aléatoires comme fonctions de tours avant d'appliquer la permutation à trappe RSA.

Soit N un module RSA de ℓ bits. Soit k_0, k_1 deux entiers. L'espace des clairs sera $\{0,1\}^{\ell-k_1-k_0}$. On considère G un oracle aléatoire de $\{0,1\}^{k_0}$ dans $\{0,1\}^{\ell-k_0}$ et H un oracle aléatoire de $\{0,1\}^{\ell-k_0}$ dans $\{0,1\}^{k_0}$. On note $\mathbf{o}^{k_1}$ la chaîne de k_1 bits nuls.

Le chiffrement de $m \in \{0,1\}^{\ell-k_1-k_0}$ par RSA-OAEP consiste à prendre $r \xleftarrow{\$} \{0,1\}^{k_0}$ et à poser

$$s := (m||\mathbf{o}^{k_1}) \oplus \mathrm{G}(r),$$
$$t := r \oplus \mathrm{H}(s).$$



On considère la chaîne de ℓ bits $s||t$ comme un entier modulo N. Le chiffré est $c := (s||t)^e \mod \mathrm{N}$.

Pour le déchiffrement, on calcule $c^d =: (s||t)$. On pose $r := t \oplus \mathrm{H}(s)$ puis on calcule $\mathrm{G}(r) \oplus s$. On vérifie que ceci est de la forme $m||\mathbf{o}^{k_1}$. Si c'est le cas on retourne m sinon $\perp$.