Fondements théoriques de la cryptographie

Hieu Phan & Philippe Guillot

13 avril 2020

Master de mathématiques.

Table des matières

Introduction

La sécurité des premiers procédés de camouflage de l'écriture, comme celui employé par Caius Julius César, résidait dans leur secret. Jusqu'à un passé assez récent, les méthodes de chiffrement étaient maintenues secrètes, dans l'espoir de retarder le temps où elles viendraient à être résolues. Une évolution majeure a eu lieu en 1883 avec la publication de l'article d'Auguste Kerckhoffs *La cryptographie militaire*. Ce dernier y énonce en effet :

> *(...) la valeur d'un système de cryptographie (...) est en raison inverse du secret qu'exige son maniement ou sa composition.*
>
> *(...) un chiffre n'est bon qu'autant qu'il reste indéchiffrable pour le maître lui-même qui l'a inventé : Ars ipsi secreta magistro.*

Pourtant, la tradition du secret a conduit les concepteurs de procédés de chiffrement à ne pas publier leurs méthodes. Les algorithmes de chiffrement utilisés dans la télévision à péage ont été volontairement maintenus secrets, et c'est encore le cas en partie pour la cryptographie militaire. Malheureusement, lorsqu'une retro-ingénierie révèle l'algorithme, des faiblesses ne manquent pas d'être découvertes puis exploitées, remettant en cause toute la sécurité du système. Un chiffre conçu dans le secret d'une officine qui ne comporte que quelques concepteurs, même très brillants, a peu de chance d'être d'une robustesse à toute épreuve. Afin de profiter des experts cryptanalystes, mathématiciens et hackers du monde entier, les nouvelles fonctions cryptographiques sont conçues dans le cadre d'une compétition ouverte où la proposition publique d'un procédé est soumises aux attaques acharnées de la communauté des cryptologues. La mise en évidence de faiblesses fait alors l'objet de correctifs, et on peut espérer qu'avec le temps, le procédé acquiert une sécurité et une confiance croissante. C'est pourquoi la conception des méthodes de chiffrement évolue vers toujours plus de transparence. Cette méthode qui consiste à révéler publiquement un algorithme cryptographique pour bénéficier d'une large évaluation de la part des cryptanalystes du monde entier constitue l'approche qualifiée de *classique* de conception des procédés cryptographiques.

C'est ainsi qu'a été conçu le *Data Encryption Standard* (DES). Le bureau des standards américain a publié un appel d'offres en 1974, auxquels ont concouru plusieurs compétiteurs. Le vainqueur a été l'algorithme conçu par Horst Feistel de l'entreprise IBM (*International Business Machines*). L'algorithme s'est montré d'une grande robustesse, les premières attaques publiées contre cet algorithme datent de 1991 et 1992, soit près de vingt ans après sa conception, sans vraiment remettre en question fondamentalement sa sécurité. Les attaques publiées nécessitent en effet une quantité de donnée qu'il est difficilement envisageable de collecter lors d'une attaque réelle. Pour remplacer cet algorithme vieillissant, un nouvel appel d'offre public a été émis en 1999, et a conduit à l'*Advanced Encryption Standard* (AES), adopté en 2001.

Mais même cette approche classique se révèle insuffisante. Combien de temps faut-il attendre pour avoir confiance ?

- Le chiffre de Vigenère, introduit par Jean-Baptiste Porta au quatorzième siècle a été considéré comme indécryptable pendant plusieurs siècles, jusqu'à ce que l'officier prussien Joseph Kasiski n'en publie une attaque en 1863.
- Une méthode de chiffrement à clé publique reposant sur le problème du sac à dos a été proposé par Benny Chor et Ronald Rivest en 1988. Il a été considéré comme sûr pendant dix ans, jusqu'à ce que Serge Vaudenay n'en publie une attaque en 1998.

— En novembre 1993, la version 1.5 du standard de chiffrement à clé publique PKCS#1 est publié par l'entreprise *RSA Laboratories*. Cinq ans plus tard, Daniel Bleichenbacher (né en 1964) publie un article décrivant comment un adversaire peut décrypter n'importe quel message par une attaque à cryptogrammes choisis [2].

Aujourd'hui, des garanties de sécurité sont exigées dans la durée. La boucle infinie des attaques et des correctifs n'est plus satisfaisante. La proposition d'un mécanisme de chiffrement doit être assortie d'une preuve de sa sécurité. La cryptologie moderne revendique une approche scientifique. Dans la préface de leur ouvrage *Introduction to Modern Cryptography* [8] publié en 2008, les auteurs Johathan Katz et Yehuda Lindell affirment :

> *Les constructions cryptographiques peuvent être prouvées sures à l'égard d'une définition de la sécurité clairement énoncée, et relativement à une hypothèse cryptographique bien définie. Ceci est l'essence de la cryptographie moderne qui a changé la cryptographie d'art en science.*

Des énoncés clairs avec des hypothèses bien définies nécessitent une formalisation des acteurs et des procédures mises en œuvre. Par exemple, un système de confidentialité met en jeu trois acteurs : l'émetteur d'un message, son destinataire et un adversaire. L'émetteur souhaite transmettre secrètement une information au destinataire. Pour cela, il chiffre le message contenant cette information selon un procédé convenu et public à l'aide d'une clé secrète partagée uniquement entre lui et son destinataire. L'adversaire, lui, ignore la clé et cherche à découvrir l'information transmise. Les objectifs et les moyens mis à disposition de ces acteurs pour accomplir ces tâches doivent être clairement précisés.

Dans de nombreuses publications, ces acteurs portent les doux noms d'Alice, Bob ou encore Ève. Contrairement à ce que ces noms laissent penser, ces acteurs ne sont pas des personnes, mais des algorithmes conçus pour atteindre un objectif précis, dans un jeu aux règles clairement énoncées. Si une preuve peut être apportée que l'adversaire ne peut extraire la moindre information sur les messages échangés malgré tous les moyens qui lui sont accordés et sans remettre en question une hypothèse largement admise, alors la sécurité du procédé sera avérée.

> *Les acteurs d'un système cryptographique sont des algorithmes qui doivent atteindre un objectif bien défini d'un jeu, avec les moyens qui leurs sont assignés.*

Ce qu'on attend d'un mécanisme de protection de l'information est qu'il soit matériellement impossible à pénétrer, qu'on puisse garantir qu'aucun adversaire réel ne puisse l'attaquer. Cette sécurité pratique dépend de la puissance de calcul dont il dispose. Mais le coût économique de cette puissance évolue. Les ordinateurs sont de plus en plus performants. De nouvelles architectures apparaissent. le modèle du calcul quantique devient de plus en plus une réalité. Toutes ces évolutions déplacent la frontière entre ce qui est réalisable et ce qui ne l'est pas. Face à ces inconnues, un mécanisme ne peut être considéré comme sûr qu'à un instant donné.

Afin de résister aux évolutions des techniques de calcul, les attaques qu'il est raisonnable d'envisager ne peuvent s'appuyer que sur des algorithmes de complexité polynomiale, c'est-à-dire sont le temps d'exécution ou la quantité de matériel exigible est bornée par un polynôme de la taille des données traitées. Si les seules attaques qui existent ont une complexité supérieure, il suffit d'augmenter légèrement la taille de la clé pour s'en prémunir et la rendre impraticable. La taille des clés est donc un paramètre crucial qui ajuste le niveau de sécurité.

> *La sécurité d'un système cryptographique est une notion asymptotique.*

En désespoir de cause, face à l'immense difficulté de mener une attaque pour découvrir les secrets échangés, un adversaire peut toujours compter sur sa chance et essayer de deviner l'information convoitée

en tirant la solution au hasard. Il n'est souvent pas raisonnable de compter sur sa bonne étoile face aux innombrables possibilités, mais lorsqu'il existe une méthode de résolution partielle, l'information résiduelle manquante peut encore être tirée au hasard et cette fois, la probabilité de succès peut devenir tout à fait significative.

Le succès d'un adversaire est une notion probabiliste.

Résoudre un chiffre doit être un problème difficile pour l'adversaire. Mais comment s'assurer de la difficulté d'un problème ? Il existe aujourd'hui des problèmes dont la difficulté est avérée, comme par exemple la factorisation des grands entiers, le calcul du logarithme discret dans un groupe cyclique ou la recherche d'un vecteur court dans un réseau. De nombreux chercheurs se penchent sur la résolution de ces problèmes mathématiques. Trouver une solution dépasse largement le domaine de la cryptographie et il est raisonnable de penser, en l'absence de solution efficace connue, que ces problèmes sont vraiment difficile.

Ainsi, prouver la difficulté de la résolution d'un chiffre peut se réduire à la résolution de l'un de ces grands problèmes dont la difficulté fait consensus. Si un adversaire peut être détourné pour, par exemple factoriser des entiers, cela montrera que la factorisation n'est pas si difficile. Du point de vue logique, cela montre que l'existence concrète d'un adversaire implique l'existence d'un algorithme de factorisation efficace. Par contraposition, l'ignorance d'un algorithme de factorisation efficace impliquera l'inexistence d'un adversaire concret.

Les preuves de sécurité sont des réductions à des problèmes réputés difficiles.

L'établissement d'une preuve ne doit cependant pas aveugler. Tout d'abord, l'existence d'un problème difficile n'est pas assurée. Aujourd'hui, un problème n'est difficile qu'en raison de notre ignorance d'une solution efficace pour le résoudre. La publication du RSA en 1978 a relancé la recherche d'algorithmes de factorisation d'entiers. Alors que les meilleurs algorithmes avaient une complexité exponentielle jusqu'aux années 1980, la décennie suivante a vu apparaître des solutions sous-exponentielles. Les progrès continueront-ils jusqu'à trouver une solution efficace ? On ne sait pas prouver aujourd'hui qu'un problème est ou n'est pas intrinsèquement difficile, et une grande question de la recherche en informatique fondamentale est la résolution de la conjecture $\mathcal{P} = \mathcal{NP}$: Est-il difficile ou non de trouver une solution à un problème dont les solutions proposées sont facilement vérifiables ? En l'absence de problème dont la difficulté est prouvée, l'édifice de la cryptographie théorique reste bâti sur du sable. Il s'effondrera dès qu'un algorithme de factorisation efficace sera découvert. Au contraire les fondations seront renforcées lorsqu'il sera prouvé que ce problème recèle en lui-même une difficulté intrinsèque.

Et pourtant, les preuves de sécurité sont loin d'être inutiles. Elles affirment qu'un adversaire ne peut réussir que s'il contourne une ou plusieurs des hypothèses sur lesquelles elles reposent. L'imagination des adversaires de la vie réelle est illimitée, et on peut compter sur leur inventivité pour trouver les moyens de faire sauter les verrous mis en place. La preuve met en avant le socle sur lequel s'appuie la sécurité. Elle est un raisonnement logique et mathématique qui n'est en rien une garantie pour les acteurs du monde réel qui utilisent les fonctions cryptographiques, mais qui informe sur ce qui fonde sa force. La cryptographie pratique reste un art où l'expertise et l'habileté du concepteur restent essentielles. Les preuves de sécurité ne sont qu'un outil supplémentaire à son service pour l'aider à concevoir des système de protection destinés à renforcer la confiance dans les communications numériques.

1 Fonctions à sens unique

Les fonctions à sens unique constituent le socle sur lequel repose pour l'essentiel l'édifice de la cryptographie théorique. De façon informelle, une fonction est à sens unique si les valeurs des images sont efficacement calculables, mais, étant donnée une valeur, il est pratiquement impossible de trouver un antécédent qui a cette valeur pour image. Cette propriété énoncée ainsi de façon informelle est celle qui est attendue en pratique, mais cette définition s'avère insuffisante pour faire opérer cette notion dans des cas concrets. Une formalisation est nécessaire.

1 Formalisation du problème

Il s'agit dans ce paragraphe de définir formellement ce qu'est une fonction à sens unique. Cette définition repose sur un jeu qui oppose deux adversaires. Le maître du jeu, appelé aussi *oracle*, met au défi un adversaire de trouver un antécédent à une valeur donnée. Les règles sont précisées ci-après :

Jeu 1.1. des fonctions à sens unique

La fonction $f : E \to F$ est donnée et connue des deux partenaires.

1. L'oracle choisit élément x aléatoire dans E avec la loi de probabilité uniforme.
2. L'oracle calcule la valeur $y = f(x)$ et transmet cette valeur y à l'adversaire.
3. Après un certain temps, l'adversaire propose un antécédent $x^\star$ de y pour la fonction f.

L'adversaire a gagné si $f(x^\star) = y$. Il a perdu dans le cas contraire.

Pour que ce jeu soit honnête pour l'adversaire, le défi y qui lui est proposé de résoudre doit être tiré au hasard selon la loi de probabilité image par la fonction f, et non pas un élément aléatoire tiré aléatoirement dans l'ensemble d'arrivée avec la probabilité uniforme.

La performance d'un adversaire se mesure à l'aune de sa *probabilité de succès*, notée $\mathrm{Pr}_{\mathrm{succes}}$.

Les ensembles de départ et d'arrivée étant des ensembles finis, il existe toujours un algorithme qui gagne à ce jeu. Il suffit de parcourir toutes les valeurs de l'ensemble de départ. On tombera immanquablement sur une valeur dont l'image vaut le défi y à résoudre. Malheureusement, lorsque les ensembles sont trop grands, la complexité de cet algorithme le rend impraticable. On devra donc limiter nos considération à des adversaire réalistes dont la complexité est bornée.

Une fonction sera dite *à sens unique* si aucun adversaire efficace n'a une probabilité raisonnable de gagner à ce jeu. Cette définition reste informelle et il reste à définir avec plus de précision ce qu'est un *adversaire efficace* et une *probabilité raisonnable*.

1.1 Fonction à sens unique idéale

Une fonction à sens unique idéale est une fonction aléatoire tirée au hasard dans l'ensemble de toutes les fonctions possibles de E vers F. Un algorithme qui renvoie la valeur $f(x)$ d'une telle fonction sur présentation du paramètre x s'appelle un *oracle aléatoire*. Les valeurs d'une telle fonction étant aléatoires et indépendantes les unes des autres, seules les valeurs déjà calculées sont accessibles. Il n'y a pas de meilleure stratégie pour d'un adversaire que d'explorer uns à uns les éléments de l'ensemble de départ,

de calculer leurs valeurs, et d'espérer de tomber sur y au bout d'un temps raisonnable. Si aucune valeur calculée ne convient, il ne reste plus, en désespoir de cause, qu'à proposer une valeur $x^\star$ tirée au hasard.

1.2 Fonction à sens unique concrète

Concrètement, on s'attend à ce qu'une fonction à sens unique soit difficile à inverser par des algorithmes qu'il est raisonnable de considérer comme réalistes. Cela est quantifié en bornant la complexité et la probabilité de succès d'un adversaire.

Définition 1.2. Fonction à sens unique concrète

Soient c, t et ε des réels positifs. Une fonction $f : E \to F$ est une fonction (c, t, ε)–à sens unique si :

1. Il existe un algorithme de complexité majorée par c qui, pour tout x de E, calcule l'image $f(x)$.
2. Tout adversaire de complexité majorée par t a une probabilité de succès inférieure ou égale à ε.

La probabilité de succès de l'adversaire est relative à une entrée y aléatoire pour la loi image par f, c'est-à-dire lorsque y est calculé comme $y = f(x)$ avec x tiré au hasard dans l'ensemble de départ avec la probabilité uniforme. Par exemple si presque tous les x ont la même valeur, la fonction f n'est pas à sens unique (voir exercice ci-après).

On attend d'une fonction (c, t, ε)–à sens unique que la valeur de t soit grande, par exemple de l'ordre de 2^ℓ, et que ε soit petit, par exemple en $1/2^\ell$, où ℓ désigne un paramètre de sécurité. Une complexité de calcul en 2^{80} est aujourd'hui considérée comme inaccessible. De même, une probabilité de succès en $1/2^{60}$ est aujourd'hui considérée comme une impossibilité. La performance d'un adversaire est donnée par le quotient t/ε.

Exercice 1.1. On suppose que la fonction $g : \{0,1\}^n \to \{0,1\}^m$ est une fonction à sens unique. On définit la fonction f par :

$$f : \ x = (x_1, \ldots, x_n) \ \mapsto \begin{cases} 0 \cdots 0 & \text{si } x_1 = 0 \\ g(x) & \text{si } x_1 = 1 \end{cases}$$

La fonction f est-elle à sens unique ?

Exemples

- Les fonctions de hachage MD5, SHA1, SHA256, SHA3 calculent une empreinte de taille fixe à partir de données de taille variable. Elles peuvent être considérées comme des fonctions à sens unique concrètes.
- Une primitive de calcul par bloc, comme le DES ou l'AES peut être utilisée comme fonction à sens unique concrète. Une telle primitive est définie comme une bijection paramétrée par une clé. Par exemple, l'AES, dans sa version 128 bits, chiffre un bloc de 128 symboles binaires avec une clé de 128 symboles binaires pour produire un cryptogramme de 128 symboles binaires :

$$\begin{array}{rccc} \mathrm{AES}_k & \{0,1\}^{128} & \to & \{0,1\}^{128} \\ & x & \mapsto & \mathrm{AES}_k(x) \end{array},$$

où k est la clé secrète appartenant à $\{0,1\}^{128}$. La fonction :

$$\begin{array}{ccc} \{0,1\}^{128} & \to & \{0,1\}^{128} \\ k & \mapsto & c = \mathrm{AES}_k(0) \end{array}$$

est une fonction à sens unique concrète. Si tel n'était pas le cas, cela signifierait qu'on peut retrouver la clé sur la base de l'observation du couple clair/cryptogramme choisi $(0, c)$, ce qui est aujourd'hui considéré comme impossible.

2 Attaques génériques

Une attaque est dite *générique* lorsqu'elle s'applique à toute fonction f sans avoir à l'expliciter. La fonction f est donnée par un sous-programme, un *oracle*, qui calcule $f(x)$ pour toute valeur x passée en paramètre. Aucune hypothèse n'est formulée quant à la structure interne de la fonction f. Les attaques génériques sont les seules applicables à un oracle aléatoire.

2.1 Borne de la force brutale

L'algorithme de force brutale consiste à calculer un très grand nombre de valeurs, et de compter sur la chance pour que la valeur à inverser soit parmi celles qui ont été calculées. Cet algorithme conduit à une première majoration des performances d'un adversaire.

Proposition 2.1. Borne de la force brutale

Soit t un réel positif. Soit f une fonction $\{0,1\}^n \to \{0,1\}^m$. On note $\mathcal{A}_t$ l'ensemble des algorithmes qui trouvent un antécédent à une valeur $y = f(x)$, pour un élément x aléatoire de E, et dont la complexité est majorée par t. Posons $\varepsilon = \max_{A \in \mathcal{A}_t}(P_{\text{succes}}(A))$. Alors :

$$\frac{t}{\varepsilon} \leq c \times 2^m.$$

Preuve Exhibons un algorithme de $\mathcal{A}_t$ dont la probabilité de succès est supérieure à $t/2^n c$. Comme ε est la probabilité de succès maximale de tels algorithmes, le résultat s'en suivra. Pour cela, considérons l'algorithme suivant :

Algorithme 2.2. Force brutale

Entrée : y, élément de l'ensemble d'arrivée dont il faut trouver un antécédent.

1. Choisir k éléments aléatoires distincts $x_1, \ldots, x_k$ dans l'ensemble de départ.
2. **pour** $i = 1$ **jusqu'à** k, calculer $y_i = f(x_i)$
3. **si** y vaut l'un des y_i **alors renvoyer** x_i
4. **sinon renvoyer** une valeur aléatoire.

La complexité de cet algorithme est majorée par le nombre maximal de calculs de valeurs de f à effectuer. Comme on effectue au plus k calculs d'une complexité chacun égal à c, on a :

$$t \leq c \times k. \tag{1.1}$$

La probabilité de succès est supérieure à la probabilité que y soit égal à l'un des y_i, elle-même supérieure à la probabilité que la valeur x tirée par l'oracle soit égale à l'un des x_i. Il est aussi possible, mais peu probable, que la valeur aléatoire convienne. De toutes façons, la probabilité de succès η de cet algorithme satisfait :

$$\eta \geq \frac{k}{2^n}. \tag{1.2}$$

En rassemblant les inégalités 1.1 et 1.2, on obtient :

$$\frac{t}{c} \leq k \leq \eta \times 2^n.$$

Le résultat s'en déduit directement. □

Si la probabilité de la loi image par f vaut la probabilité uniforme sur l'ensemble d'arrivée, ce qui est sensiblement le cas des fonctions de hachage cryptographique standards, on peut établir

une égalité similaire sur le nombre de symboles de l'ensemble d'arrivée :

$$\frac{t}{\varepsilon} \leq c \times 2^m$$

Exemple d'application. L'unité de complexité est le temps de calcul de la fonction f, soit $c = 1$. On veut qu'un adversaire qui réussisse avec une probabilité de succès inférieure à $1/2^{60}$ ait une complexité supérieure à 2^{80}. Combien de symboles binaires doivent comporter les valeurs de f ?

Les conditions posées se traduisent par $t \geq 2^{80}$ et $\varepsilon \leq 1/2^{60}$. Il faut donc :

$$2^m \geq \frac{t}{c\varepsilon} \geq \frac{2^{80}}{1/2^{60}} = 2^{140}.$$

Les valeurs de la fonction doivent avoir au moins 140 symboles binaires.

2.2 Borne du compromis temps-mémoire

Un algorithme générique notablement plus efficace que la force brutale a été publié en 1980 par Martin Hellman (1945 -) [6]. L'algorithme fonctionne en deux temps. Le premier temps consiste en un long précalcul avec mémorisation des résultats, indépendant de la valeur à inverser. Le second temps exploite les résultats du précalcul pour chercher un antécédent à une valeur donnée. Le précalcul étant effectué une fois pour toutes, il n'est n'est pas comptabilisé dans la complexité de recherche d'antécédent.

Comme cet algorithme repose sur des compositions successives de la fonction f, il n'est applicable que si la fonction admet les mêmes ensembles de départ et d'arrivée. Il faut aussi supposer que la fonction f est bijective.

Proposition 2.3. Borne du compromis temps-mémoire

Avec les mêmes notation que dans la proposition 1, on a :

$$\frac{t^2}{\varepsilon} \leq c^2 \times 2^n.$$

Preuve On considère l'algorithme suivant qui comporte deux phases : un précalcul, indépendant de la valeur à inverser, et un calcul fonction de la valeur y à inverser :

Algorithme 2.4. compromis temps-mémoire

1. **Précalcul.** Considérer k valeurs aléatoires $x_1, \ldots x_k$.
   1. **pour** $i = 1$ jusqu'à k, calculer les valeurs $z_i = f^k(x_i)$.
   2. Mémoriser les couples (x_i, z_i) dans une table.
2. **Calcul.** L'entrée est une valeur $y \in \{0,1\}^n$ à inverser.
   1. Pour $j = 1$ jusqu'à $k-1$, calculer les itérés $y_j = f^j(y)$.
   2. Dès que l'un des y_j vaut l'un des z_i, alors renvoyer la valeur $x = f^{k-j-1}(x_i)$.
   3. Si aucun des y_j n'est égal à l'un des z_i, alors renvoyer une valeur x aléatoire dans $\{0,1\}^n$.

Dans le cas où $z_i = y_j$, on a $f^k(x_i) = f^j(y)$, par conséquent, f étant bijective, $y = f^{-j}(y)$ et $f(x) = f^{k-j}(x_i) = f^{-j}(z_i) = y$. Dans ce cas, la probabilité de succès vaut 1.

Dans le second cas, l'algorithme réussit lorsque le hasard tombe sur la bonne solution, soit avec une probabilité égale à $1/2^n$.

L'algorithme réussit toujours s'il existe un élément commun aux deux listes (z_i) et (y_j). Les valeurs des x_i étant aléatoires, la probabilité de cet événement est donnée par le paradoxe des anniversaires.

Quand n tend vers l'infini, cette probabilité tend vers $k^2/2^n$. On peut considérer que la probabilité de succès de l'algorithme satisfait l'inégalité $\varepsilon \geq k^2/2^n$.

La complexité du précalcul n'est pas comptabilisée, car celui-ci est effectué une fois pour toutes. La complexité de la phase de calcul est :

$$t = \underbrace{c \times j}_{\text{calcul des } y_1, \ldots, y_j} + \underbrace{c \times (k-j-1)}_{\text{calcul de } x = f^{k-j-1}(x_i)} = c \times k.$$

La complexité en mémoire est de $2k$ vecteurs de n composantes binaires.

On en déduit :

$$\frac{t^2}{c^2} = k^2 \leq \varepsilon \times 2^n,$$

donc

$$\frac{t^2}{\varepsilon} \leq c^2 \times 2^n.$$

□

Exercice 1.2. L'unité de complexité est le temps de calcul d'une valeur de la fonction f. Combien de symboles binaires doivent comporter les valeurs de f pour satisfaire les exigences de sécurité $t \geq 2^{80}$ et $\varepsilon \leq 1/2^{60}$?

Encadré 2.5. Le paradoxe des anniversaires

Soit $(z_1, \ldots, z_k)$ une liste de k termes aléatoires et $(y_1, \ldots, y_k)$, une liste quelconque de k termes choisis dans un ensemble de 2^n éléments. Il s'agit de déterminer la probabilité que ces deux listes aient un terme commun. Soit q la probabilité de l'événement contraire, c'est-à-dire la probabilité qu'aucun des y_j ne soit égal à l'un des z_i. Cela signifie que tous les z_i sont dans l'ensemble des $2^n - k$ éléments différents des termes de la liste (y_j). On a donc

$$q = \left(\frac{2^n - k}{2^n}\right)^k = \left(1 - \frac{k}{2^n}\right)^k.$$

En développant ce produit au premier ordre, et en considérant que $k/2^n$ est petit devant son carré, on trouve que q est équivalent, au voisinage de l'infini, à :

$$1 - k\left(\frac{k}{2^n}\right) = 1 - \frac{k^2}{2^n},$$

d'où on déduit la probabilité de collision $1 - q$ qui est équivalente à $k^2/2^n$.

3 Exemples pratiques

On ne sait pas si les fonctions à sens unique existent réellement. Un des défis de la cryptographie théorique est de montrer ou d'infirmer leur existence. Il existe cependant des fonctions faciles à calculer et pour lesquelles on ne dispose pas aujourd'hui d'algorithme efficace pour les inverser. Cela ne signifie pourtant pas qu'il s'agit réellement de fonctions à sens unique. Il est possible que les recherches futures permettent de découvrir un algorithme efficace d'inversion. Une autre piste de la recherche est de prouver qu'un tel algorithme n'existe pas, mais cela impliquerait que $\mathcal{P} \neq \mathcal{NP}$, ce qui est l'une des grandes conjectures du siècle.

3.1 La multiplication

L'ensemble de départ est l'ensemble des couples de nombres premiers. $E = \{(p, q) \mid p, q \text{ premiers}\}$.

l'ensemble d'arrivée est l'ensemble $\mathbb{N}$ des entiers naturels.

La multiplication consiste à faire le produit des paramètres :

$$\text{MUL} : \begin{array}{ccc} E & \to & \mathbb{N} \\ (p,q) & \mapsto & p \times q \end{array}.$$

Il existe des algorithmes efficaces pour calculer le produit de deux nombres. L'algorithme scolaire, par exemple, a une complexité quadratique en la taille des entiers à multiplier. Les meilleurs algorithmes de multiplication connus reposent sur la transformation de Fourier discrète et ont une complexité quasi linéaire.

Par contre, les meilleurs algorithmes de factorisation connus à ce jour ont une complexité sur-polynomiale. Étant donné un produit de deux nombres premiers, il est aujourd'hui difficile de trouver les facteurs. Lorsque les deux nombres premiers ont à peu près la même taille, et lorsque le produit dépasse quelques centaines de chiffres décimaux, on estime qu'il est pratiquement impossible de trouver les facteurs. La multiplication est aujourd'hui considérée comme une fonction à sens unique.

En 2018, factoriser un entier de 1 024 symboles binaires égal au produit de deux grand nombres premiers est possible mais nécessite des moyens énormes, accessible uniquement aux grandes entreprises, aux états ou aux grandes institutions. La factorisation des entiers de 2 048 symboles binaires est considéré comme impossible.

Exercice 1.3. Le meilleur algorithme de factorisation connu aujourd'hui s'appelle le *crible du corps de nombres*. La complexité estimée pour factoriser un entier n est estimée à :

$$L(n) = \exp\big(k(\ln n)^{1/3}(\ln\ln b)^{2/3}\big),$$

avec $k = 64/9$. On suppose qu'aujourd'hui, factoriser un entier de 200 chiffres binaires prend une heure. Quel temps faut-il pour factoriser un entier de 300 chiffres binaires sur la même machine ?

Exercice 1.4. La fonction MUL reste-t-elle à sens unique si on ne suppose pas que les entiers p et q sont premiers ?

3.2 La fonction RSA

Soient p et q deux entiers premiers assez grands pour que la factorisation de leur produit soit considéré comme pratiquement impossible. On pose $n = p \times q$ et $\lambda(n) = \text{ppcm}(p-1, q-1)$. Soit e un entier premier avec $\lambda(n)$. La fonction RSA est la suivante :

$$\text{RSA} : \begin{array}{ccc} \mathbb{Z}/n\mathbb{Z} & \to & \mathbb{Z}/n\mathbb{Z} \\ x & \mapsto & x^e \end{array}$$

Il est largement admis qu'inverser cette fonction, c'est-à-dire calculer $\sqrt[e]{y}$ modulo n est pratiquement impossible sans la connaissance de la factorisation de l'entier n. Par contre, si cette factorisation est connue, alors le calcul de l'image réciproque de y est donné par $x = y^d$, où d est l'inverse de e modulo $\lambda(n)$.

Exercice 1.5. Montrer que la connaissance de l'exposant d permet de factoriser l'entier n.

Connaître la factorisation de l'entier n permet d'inverser la fonction RSA, mais on ne sait pas si la réciproque est vrai, c'est-à-dire si savoir inverser la fonction RSA permet de factoriser l'entier n.

3.3 La fonction exponentielle sur un corps fini

Soit p un nombre premier et $\mathbb{F}_p$ le corps fini à p éléments. On sait que le groupe multiplicatif de ce corps est un groupe cyclique. Soit g un générateur de ce groupe. La fonction exponentielle en base g sur $\mathbb{F}_p$ est par définition :

$$\text{EXP}_g : \begin{array}{ccc} \mathbb{F}_p^\star & \to & \mathbb{F}_p^\star \\ x & \mapsto & g^x \end{array}$$

Pour un élément $y \in \mathbb{F}_p^\star$, l'élément x tel que $y = g^x$ s'appelle le *logarithme discret de y en base g*. Le calcul des logarithmes discrets dans un corps premier reste un problème difficile.

Une variante de cette fonction est la multiplication dans le groupe des points d'une courbe elliptique. Soit $(\mathcal{E}, +)$ le groupe des points d'une courbe elliptique définie sur un corps fini, soit P un générateur de ce groupe et soit p l'ordre de ce groupe. La multiplication par le point P est définie par :

$$\text{MUL}_P : \begin{array}{ccc} \mathbb{Z}/p\mathbb{Z} & \to & \mathcal{E} \\ n & \mapsto & n \cdot P \end{array}$$

La fonction inverse, qui consiste, pour un point R de $\mathcal{E}$, à trouver le multiple n tel que $R = n \cdot P$, est un problème difficile. Cet entier n s'appelle aussi le *logarithme discret de R dans $\mathcal{E}$ en base P*.

3.4 La fonction carré modulo un produit de deux nombres premiers

Soient p et q deux nombres premiers distincts tels que la factorisation du produit $n = p \times q$ soit en pratique impossible. La fonction carré dans $\mathbb{Z}/n\mathbb{Z}$ est par définition :

$$\text{SQR} : \begin{array}{ccc} \mathbb{Z}/n\mathbb{Z} & \to & \mathbb{Z}/n\mathbb{Z} \\ x & \mapsto & x^2 \end{array}$$

L'exercice suivant montre que, tant que la multiplication sera une fonction à sens unique, c'est-à-dire tant qu'il sera difficile de factoriser les grands entiers, la fonction carré modulo n restera une fonction à sens unique.

Exercice 1.6. Soit n un produit de deux nombres premiers p et q. Démontrer que savoir factoriser n avec un algorithme de complexité polynomiale équivaut à savoir inverser la fonction carré modulo n avec un algorithme de complexité polynomiale.

4 Fonction asymptotiquement à sens unique

En cryptographie, la sécurité repose sur une clé dont la taille est cruciale pour en déterminer le niveau. Plus la clé est longue, et plus a sécurité attendue est élevée. Il est admis qu'une attaque n'est raisonnablement envisageable que si l'algorithme qui la conduit a une complexité bornée par un polynôme de cette taille. Dans le cas contraire, il suffit d'augmenter légèrement la taille de la clé pour se mettre à l'abri des attaques. Mais si l'algorithme a une complexité polynomiale, alors l'augmentation de la valeur du paramètre pour résister à l'attaque compromettrait gravement le temps de chiffrement et de déchiffrement. Par ailleurs, faire tourner plusieurs adversaires en parallèle pourrait mener l'attaque à son terme.

La sécurité, comme fonction du paramètre de sécurité que constitue la taille des clés est une notion asymptotique. La complexité de calcul, la complexité d'attaque, comme la probabilité de succès s'évaluent comme fonction du paramètre de sécurité et comment ces fonctions se comportent lorsque ce paramètre croît.

Encadré 4.1. Fonction négligeable

La notion de fonction négligeable sert à qualifier la probabilité de succès d'un adversaire lorsque la valeur du paramètre de sécurité tend vers l'infini. On s'attend à ce que cette probabilité elle tende vers zéro, mais on exige encore davantage. Une fonction est négligeable si elle décroit vers zéro plus vite que l'inverse de tout polynôme.

Définition 4.2. fonction négligeable

Une fonction $\varepsilon : \mathbb{N} \to \mathbb{R}$ est dite négligeable si :

$$\forall k \in \mathbb{N} \quad \exists N \in \mathbb{N} \quad \forall n \geq N \quad |\varepsilon(n)| < \frac{1}{n^k}$$

Par exemple, la fonction $n \mapsto 1 : \sqrt{2^n}$ est négligeable, mais la fonction $n \mapsto 1/x^{1000}$ ne l'est pas.

Proposition 4.3.

L'ensemble des fonctions négligeables est stable par addition, par multiplication et par élévation à toute puissance.

La preuve de cette proposition est directe et est laissée en exercice.

Il n'y a pas de problème à se restreindre aux messages qu'on peut exprimer avec l'alphabet binaire. On note $\{0,1\}^\star$ le monoïde libre sur l'alphabet $\{0,1\}$. Il s'agit par définition de l'ensemble des mots de toute longueur constitués de 0 et de 1, muni de l'opération associative que constitue la concaténation.

$$\{0,1\}^\star = \bigcup_{n\in\mathbb{N}} \{0,1\}^n.$$

L'élément neutre pour la concaténation est le mot vide.

Une fonction à sens unique asymptotique est définie sur $\{0,1\}^\star$ pour des mots de toute longueur. Deux propriétés sont exigées d'une telle fonction, comme énoncé dans cette définition : elle doit être facile à calculer et difficile à inverser.

Définition 4.4. fonction asymptotiquement à sens unique

Une fonction $f : \{0,1\}^\star \to \{0,1\}^\star$ est une fonction asymptotiquement à sens unique si les deux conditions suivantes sont remplies :

1. Il existe un algorithme de complexité polynomiale qui calcule $f(x)$ pour toute valeur du paramètre $x \in \{0,1\}^\star$.
2. Tout adversaire polynomial contre la fonction f au jeu des fonctions à sens unique a une probabilité de succès négligeable.

Le paramètre n de sécurité pour l'adversaire est la taille de l'élément $x \in \{0,1\}^n$ choisi par l'oracle du jeu des fonctions à sens unique. La seconde condition s'exprime donc en énonçant que la fonction :

$$n \mapsto \Pr(A(y) = x^\star \text{ et } f(x^\star) = y),$$

est une fonction négligeable.

La probabilité est relative au choix aléatoire par l'oracle de x dans l'ensemble $\{0,1\}^n$ avec la loi uniforme pour définir l'entrée y de l'adversaire, égale à $f(x)$. L'adversaire est qualifié de polynomial si sa complexité est borné par un polynôme en n, où n est la taille de x et non pas celle de son entrée y.

Comme le calcul de la valeur d'une fonction à sens unique est un problème polynomial, retrouver un antécédent est un problème $\mathcal{NP}$. Par conséquent, une fonction à sens unique asymptotique ne peut exister que si $\mathcal{P} \neq \mathcal{NP}$. On ne sait toujours pas si les fonctions à sens unique existent vraiment. Si leur existence est vraisemblable, mais ne peut toujours pas être prouvée.

Encadré 4.5. Une fonction difficile à inverser dans le pire des cas qui n'est pas à sens unique

Si $\mathcal{P} \neq \mathcal{NP}$, un problème $\mathcal{NP}$–complet est un problème difficile à résoudre dans le pire des cas, mais une fonction est à sens unique si elle est difficile à inverser dans le cas moyen, c'est-à-dire pour un élément y choisi selon la loi image de la loi uniforme par f. Si on construit une fonction à partir d'un problème $\mathcal{NP}$–complet, elle sera difficile à inverser dans le pire des cas, mais pourra être facile à inverser dans le cas moyen.

Prenons par exemple le problème 3COL du coloriage d'un graphe avec trois couleurs.

Définition 4.6. Graphe

Un graphe de n sommets est une matrice symétrique carrée de dimension n sont les coefficients appartiennent à l'ensemble $\{0,1\}$.

Chaque numéro de composante représente un numéro de sommet et les coefficients égaux à 1 représentent des arrêtes entre deux sommets. Deux sommets a et b sont adjacents dans le graphe $G = (g_{ij})$ si le coefficient g_{ab} de la matrice G vaut 1.

Définition 4.7. Coloriage avec trois couleurs

Un coloriage d'un graphe avec trois couleurs est une application c de l'ensemble $\{1,\dots,n\}$ vers l'ensemble $\{1,2,3\}$ tel que deux sommets adjacents n'ont pas la même image, c'est-à-dire :

$$\forall i, i \in \{1,\dots,n\},\ g_{ij} = 1 \Rightarrow c(i) \neq g(j).$$

L'image d'un sommet représente la couleur dans laquelle il est colorié.

Rappelons, en les admettant, les résultats suivants :

— Si on sait qu'un graphe est coloriable avec trois couleurs, il est difficile de trouver un coloriage.

— Si on tire un graphe G au hasard et une application de coloriage c, il est hautement improbable que c soit un coloriage de G.

Notons $\mathcal{G}_n$ l'ensemble des graphes de n sommets, et considérons la fonction suivante :

$$f : \begin{array}{ccc} \mathcal{G}_n \times \{1,2,3\}^n & \to & \mathcal{G}_n \times \{0,1\} \\ (G,c) & \mapsto & \begin{cases} (G,1) & \text{si } c \text{ colorie } G \\ (G,0) & \text{sinon} \end{cases} \end{array}$$

Pour un couple (G,c) aléatoire, il est très probable que l'application c ne colorie pas le graphe G et donc la valeur $f(G,c)$ vaut très probablement $(G,0)$. Dans ce cas, tout couple (G,c'), avec c' quelconque, est très probablement un antécédent de $(G,0)$. Les seuls cas où trouver un antécédent sera difficile sera lorsqu'il faudra trouver un antécédent à l'élément $(G,1)$. Une réponse au hasard conduira sûrement à un échec, mais cela ne surviendra que très rarement.

La fonction f est facilement inversible en moyenne, ce n'est pas une fonction à sens unique. Par contre, elle est difficile à inverser dans le pire des, cas, c'est-à-dire lorsqu'il s'agit de trouver un antécédent à un élément de la forme $(G,1)$ où G est coloriable avec trois couleurs.

5 Familles de fonctions à sens unique

La notion de fonction à sens unique asymptotique est théorique. Les fonctions à sens unique utilisées en pratique, comme les fonctions MUL, RSA, EXP et SQR, sont en réalité des familles de fonctions à sens unique. Ce sont des fonctions indexées par un indice appartenant à un ensemble I. Chaque indice définit un domaine sur lequel la fonction opère ainsi qu'un paramètre de sécurité qui quantifie la sécurité, c'est-à-dire l'effort actuel pour inverser la fonction.

Présentons tout d'abord les familles candidates pour être des fonctions à sens unique.

La famille $(MUL_n)_{n\in I}$

L'ensemble des indices I est l'ensemble des entiers naturels $\mathbb{N}$. Pour un entier n, le domaine de la fonction MUL_n est l'ensemble des couples (p,q) de deux nombres premiers dont la taille ne dépasse pas n chiffres binaires. La difficulté d'inverser la fonction MUL_n dépend de taille du produit, donc dépend directement de la valeur de l'entier n. Le paramètre de sécurité est l'entier n lui-même.

La famille $(RSA_{(m,e)})_{(m,e)\in I}$

L'ensemble d'indices I est l'ensemble des couples (m, e), où m est un entier égal au produit de deux nombres premiers distincts p et q, et e est un entier premier avec $p-1$ et $q-1$.

Le domaine de la fonction $RSA_{(m,e)}$ est l'ensemble $\mathbb{Z}/m\mathbb{Z}$.

Le paramètre de sécurité est le nombre de chiffres binaires de l'entier m.

La famille $(EXP_{(p,g)})_{(p,g)\in I}$

L'ensemble des indices I est l'ensemble des couples (p, g), où p est un nombre premier, et où g est un générateur du groupe multiplicatif $\mathbb{Z}/p\mathbb{Z}^*$. Le domaine de la fonction $EXP_{p,g}$ est l'ensemble $\{1, \ldots, p-1\}$.

Le paramètre de sécurité de cette fonction est le nombre de chiffres binaires de l'entier p.

La famille $(SQR_m)_{m\in I}$

L'ensemble d'indices I est l'ensemble entiers m égaux au produit de deux nombres premiers distincts. Le domaine de la fonction SQR_m est l'ensemble $\mathbb{Z}/m\mathbb{Z}$.

Le paramètre de sécurité de cette fonction est le nombre de chiffres binaires de l'entier m.

Pour que ces familles soient utilisables en pratique, il est nécessaire de savoir produire des indices de la famille qui respectent une taille qui permette d'assurer une certaine sécurité. Il est par exemple admis que la sécurité de la fonction $RSA_{(m,e)}$ n'est assurée que si l'entier m a une taille d'au moins 1 024 symboles binaires.

Il est également nécessaire de pouvoir produire pratiquement des éléments du domaine et d'en calculer la valeur par des algorithmes efficaces.

Ces remarques conduisent à la définition formelle de la notion de famille de fonctions à sens unique. Nous montrerons ensuite que cette notion est finalement équivalente à celle de fonction à sens unique asymptotique.

Définition 5.1. Famille de fonctions à sens unique

Soit I un ensemble d'indices, qu'on peut supposer inclus dans l'ensemble $\{0,1\}^\star$. Soit $\mathcal{F} = (f_i)_{i\in I}$ une famille de fonctions d'un domaine D_i vers l'ensemble $\{0,1\}^*$. On dit que $\mathcal{F}$ est une famille de fonctions à sens unique si les trois conditions suivantes sont remplies :

1. $\mathcal{F}$ est efficacement échantillonnable, c'est-à-dire :
   a) Il existe un algorithme probabiliste efficace $\mathcal{S}_1$, dont l'entrée est un paramètre de sécurité $n \in \mathbb{N}$, et qui produit un indice i, aléatoire dans I tel que les éléments du domaine D_i ont une taille bornée par un polynôme $\ell(n)$ en n.
   b) Il existe un algorithme probabiliste efficace $\mathcal{S}_2$, qui pour l'entrée $i \in I$ générée par l'algorithme $\mathcal{S}_1$, produit un élément x aléatoire dans D_i.
2. Les valeurs des fonctions f_i sont efficacement calculables, c'est-à-dire il existe un algorithme efficace qui, à partir de tout élément $i \in I$ et de tout élément $x \in D_i$, calcule la valeur $f_i(x)$.
3. Les fonctions f_i sont difficiles à inverser, ce qui signifie que pour tout élément y égal à $f_i(x)$, où i est le résultat de l'algorithme S_1 pour l'entrée n, et x est le résultat de l'algorithme S_2 pour l'entrée i, la probabilité de succès de tout algorithme efficace en n pour trouver un antécédent de y est négligeable en n.

Un algorithme probabiliste est un algorithme qui produit un résultat aléatoire pour une loi de probabilité donnée. On peut toujours considérer un tel algorithme comme un algorithme déterministe avec un paramètre supplémentaire constitué d'une chaîne binaire aléatoire en fonction de laquelle l'élément aléatoire est produit.

Par exemple, pour produire une clé RSA de 1 024 symboles binaires, il faut produire deux nombres premiers distincts dont la somme des tailles fait précisément 1 024 symboles binaires. Cela peut se faire à partir d'un germe aléatoire constitué d'une chaîne binaire de cette taille, partagée en deux sous-chaînes de tailles sensiblement égales. Ces deux sous-chaînes sont alors interprétées comme l'écriture binaire d'un entier qui est le paramètre d'une fonction `next_prime` qui renvoie l'entier premier suivant.

Pour des raisons d'efficacité, l'exposant e est imposé, par exemple égal à 3 ou au nombre premier 2^{16+1}, ce qui est le cas dans la norme EMV *Europay, Mastercard, Visa* des cartes bancaires. Dans ce cas les nombres premiers p qui composent le module public doivent être choisis tels que $p-1$ est premier avec cet exposant.

Pour générer un indice (p, g) de la fonction $EXP_{p,g}$, il est nécessaire de produire un nombre premier p d'une taille fixée et un générateur du groupe multiplicatif $\mathbb{Z}/p\mathbb{Z}^\star$. Tester si un entier g est générateur nécessite de connaître les facteurs premiers de $p-1$. La taille utilisée pour p empêche en pratique de compter sur un algorithme de factorisation pour résoudre le problème. Il est plus raisonnable de chercher à produire un entier r dont la factorisation est *a priori* connue, et constituée au moins d'un grand nombre premier q, jusqu'à obtenir un entier $p = r + 1$ premier. La contrainte sur le grand facteur premier de $p-1$ est nécessaire pour rendre plus difficile le calcul du logarithme discret dans $(\mathbb{Z}/p\mathbb{Z}^\star, \times)$.

Exercice 1.7. Proposer un algorithme qui génère un indice (p, g) pour la fonction exponentielle. L'entier premier p doit être de 1 024 chiffres binaires et avoir un facteur premier d'au moins 256 chiffres binaires.

Théorème 5.2. Equivalence des deux notions

Il existe une fonction à sens unique asymptotique si et seulement si il existe une famille de fonctions à sens unique.

Preuve La preuve de ce théorème est constructive. Soit f une fonction qu'on suppose être à sens unique asymptotique. On construit à partir de f une famille de fonctions à sens unique ainsi :

- L'ensemble d'indices est $I = \{0,1\}^\star$;
- Pour $i \in I$, le domaine D_i est l'ensemble des mots qui ont la même longueur que i, c'est-à-dire $D_i = \{0,1\}^{|i|}$.
- L'algorithme $\mathcal{S}_1$ produit, pour l'entrée $n \in \mathbb{N}$, un mot aléatoire i de longueur n.
- L'algorithme $\mathcal{S}_2$ produit, pour l'entrée $i \in I$, un mot binaire aléatoire de même longueur que i.
- Pour tout indice $i \in I$, la fonction f_i est définie par :

$$f_i : \begin{array}{ccc} D_i & \to & \{0,1\}^\star \\ x & \mapsto & f(x) \end{array}$$

Montrons, sous l'hypothèse que la fonction f est à sens unique asymptotique, que la famille $(f_i)_{i\in I}$ est une famille de fonctions à sens unique.

Tout d'abord, la fonction f est facile à calculer, il en est donc ce même pour chaque f_i. Ensuite, montrons que chaque f_i est difficile à inverser. Pour cela supposons le contraire et supposons l'existence d'un adversaire A qui réussit à inverser f_i avec une probabilité de succès non négligeable. Cet adversaire peut inverser la fonction f avec le même succès, ce qui est contraire à l'hypothèse.

Réciproquement, soit $\mathcal{F} = (f_i)_{i\in I}$ une famille de fonctions à sens unique. Construisons à partir d'elle une fonction asymptotiquement à sens unique f.

Rappelons qu'un algorithme qui rend un résultat aléatoire peut se modéliser comme un algorithme déterministe qui prend en paramètre un mot binaire aléatoire servant de germe à la génération de l'aléa nécessaire à son calcul.

Pour un élément z de $\{0,1\}^\star$, on définit la valeur de $f(z)$ ainsi :

On coupe le mot binaire z en deux mots disjoints de taille convenable pour alimenter respectivement $\mathcal{S}_1$ et $\mathcal{S}_2$ en chaîne aléatoire. Soit $z = (r, s)$.

On utilise le premier terme r comme chaîne aléatoire pour l'algorithme $\mathcal{S}_1$, et on utilise la deuxième moitié s comme chaîne aléatoire pour l'algorithme $\mathcal{S}_2$. Ces algorithmes fournissent de manière déterministe un indice $i = \mathcal{S}_1(r) \in I$, puis un élément $x = \mathcal{S}_2(s, i) \in D_i$. On pose finalement :

$$f(r, s) = (f_i(x), i).$$

Il reste à montrer que la fonction f est à sens unique. Elle est facile à calculer, car les algorithmes $\mathcal{S}_1$ et $\mathcal{S}_2$ sont efficaces. Montrons qu'elle est difficile à inverser. Pour cela, en vue d'une contradiction, supposons le contraire. Cela signifie qu'il existe un algorithme efficace A qui, sur une entrée $(y, i) \in \{0,1\}^\star \times I$, trouve un élément (r, s) vérifiant $i = \mathcal{S}_1(r)$ et $f(r, s) = f_i(y)$. Soit B l'algorithme qui, à partir du résultat de A, calcule $x = \mathcal{S}_2(s, i)$. Cette valeur est calculable en temps polynomial, et, en cas de succès de A, est un antécédent x de y par f_i. L'algorithme B inverse donc efficacement la fonction f avec un succès supérieur à celui de A, ce qui montre que sous cette hypothèse, la famille (f_i) n'est pas à sens unique. □

Exercice 1.8. Soit $g : E \longrightarrow E$ une fonction à sens unique et a un élément particulier de E.

1. Démontrer que la fonction $f : (x, y) \mapsto \big(f(y), a\big)$ est à sens unique.
2. En déduire que si f est à sens unique, la composée $f \circ f$ n'est pas toujours à sens unique ?
3. Soit f une fonction à sens unique $E \to E$. On suppose que f est une bijection. Démontrer que $f \circ f$ est à sens unique.

Exercice 1.9. On dit qu'une fonction $\{0,1\}^\star \to \{0,1\}^\star$ est *à longueur régulière* si :

$$\forall x, y \in \{0,1\}^\star, \quad |x| = |y| \Rightarrow \big|f(x)\big| = \big|f(y)\big|,$$

où la notation $|x|$ désigne le nombre de symboles binaires du mot x. Montrer que s'il existe une fonction à sens unique, alors il existe une fonction à sens unique à longueur régulière.

Exercice 1.10. On dit qu'une fonction $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ *conserve les longueurs* si, pour tout $x \in \{0,1\}^*$, on a $\big|f(x)\big| = |x|$. Montrer que s'il existe une fonction à sens unique, alors il existe une fonction à sens unique qui conserve les longueurs.

6 Prédicat difficile

Le moindre exigible d'une fonction à sens unique est que la valeur ne révèle pas au moins une certaine information sur l'entrée. Ce qu'on appelle un prédicat difficile (*hard core predicate*) est précisément une telle information qu'il est difficile de reconstituer à partir de la valeur. Un résultat fondamental de la cryptologie théorique est l'existence d'un tel prédicat, existence affirmée par le théorème de Goldreich-Levin montré dans cette section.

Un prédicat difficile d'une fonction à sens unique permet de construire les générateurs de pseudo-aléa, qui serviront pour définir des algorithmes de chiffrement et d'authentification.

Encadré 6.1. Avantage d'un adversaire

La probabilité de succès n'est pas toujours le bon paramètre pour évaluer les performance d'un adversaire. Par exemple, si la réponse est binaire et équidistribuée, un adversaire qui répond au hasard a une probabilité de succès de $1/2$ qui n'est pas négligeable. Il est plus judicieux d'employer un autre critère : celui de l'*avantage*.

Définition 6.2. Avantage

Soit A un algorithme supposé trouver une réponse correcte sur une entrée x. Soit B_x l'ensemble des réponses correctes possibles. L'avantage de A est par définition la quantité :

$$\mathrm{Av}(A) = \Big|\underbrace{\Pr\big(A(x) \in B_x\big)}_{(1)} - \underbrace{P\big(A(x) \in B_{x^\star}\big)}_{(2)}\Big| \times \underbrace{\frac{1}{\Pr\big(A(x) \notin B_{x^\star}\big)}}_{(3)},$$

où $x^\star$ est un élément aléatoire.

Le premier terme (1) est la probabilité de succès de A sur l'entrée x, notée $\mathrm{P}_{\text{succes}}(A)$.

Le deuxième terme (2) est la probabilité que la réponse de A sur l'entrée x soit la réponse attendue, non pas pour x, mais pour un élément $x^\star$ aléatoire indépendant de x.

Le deuxième facteur (3) est un coefficient de normalisation qui donne un avantage égal à 1 pour un adversaire qui répond toujours correctement, c'est-à-dire pour lequel la probabilité de succès $\Pr\big(A(x) \in B_x\big)$ vaut 1.

Avec cette définition, un adversaire qui répond au hasard a toujours un avantage nul.

Dans le cas où la réponse est binaire et la probabilité de bonne réponse équidistribuée, en posant b la bonne réponse, $\widehat{b} \in \{0,1\}$ la réponse de l'algorithme A, et $b^\star$ un élément aléatoire de $\{0,1\}$ indépendant de b et $\widehat{b}$, l'avantage s'exprime :

$$\begin{aligned}\mathrm{Av}(A) &= \Big|\Pr(\widehat{b}=b) - \Pr(\widehat{b}=b^\star)\Big| \times \frac{1}{\Pr(\widehat{b}\neq b^\star)} \\ &= \Big|\mathrm{P}_{\text{succes}}(A) - \frac{1}{2}\Big| \times 2 \\ &= \big|2\mathrm{P}_{\text{succes}}(A) - 1\big|,\end{aligned}$$

ou, ce qui est équivalent :

$$\mathrm{P}_{\text{succes}}(A) = \frac{1}{2} + \frac{\mathrm{Av}(A)}{2}.$$

Exercice 1.11. On suppose que $f : \{0,1\}^\star \to \{0,1\}^\star$ est une fonction à sens unique. On pose φ la fonction suivante :

$$\varphi : \begin{array}{ccc}\{0,1\}^n \times \mathbb{N} & \to & \{0,1\}^\star \times \{0,1\} \times \mathbb{N} \\ (x,i) & \mapsto & \big(f(x), x_i, i\big)\end{array}.$$

1. Démontrer que φ est une fonction à sens unique.
2. Pour tout entier naturel j, définir un algorithme A_j qui, à partir d'une valeur $\varphi(x) = (y,\eta,i) \in \{0,1\}^\star \times \{0,1\}\times\mathbb{N}$, obtenue à partir d'une valeur de x aléatoire, trouve la valeur x_j avec un avantage non négligeable.

L'exercice 11 ci-dessus montre qu'il peut exister des fonctions à sens unique qui ne masquent aucune des composantes de leur paramètre. Un prédicat difficile d'une fonction à sens unique n'est pas nécessairement une composante, mais plus généralement une fonction booléenne de l'entrée qui représente une information de l'entrée supposée être inaccessible connaissant uniquement la valeur de la fonction.

Définition 6.3. prédicat difficile

Soit $f : \{0,1\}^\star \to \{0,1\}^\star$ une fonction. La fonction booléenne $p : \{0,1\}^\star \to \{0,1\}$ est un *prédicat difficile* pour f si les deux conditions suivantes sont remplies :

- Pour tout $x \in \{0,1\}^\star$, la valeur de $p(x)$ est calculable en temps polynomial.
- Tout algorithme polynomial A ne peut calculer $p(x)$ à partir de $y = f(x)$, pour une valeur aléatoire de x, qu'avec un avantage négligeable.

En d'autres termes, la fonction

$$n \mapsto \Big|2\Pr(A(y) = p(x)) - 1\Big|,$$

où $y = f(x)$ pour un élément aléatoire $x \in \{0,1\}^n$, est une fonction négligeable.

L'avantage d'un adversaire contre un prédicat supposé difficile d'une fonction est formalisé par le jeu du prédicat difficile défini ci-après. Ce jeu oppose un oracle à un challenger. Les joueurs connaissent une fonction $f : E \to F$ et un prédicat $p : E \to \{0,1\}$. Le but de l'adversaire est de trouver l'information $p(x)$ sur x à partir d'une valeur $y = f(x)$ qui lui est donnée. Les règles de ce jeu sont précisée ci-après :

Jeu 6.4. du prédicat difficile

1. L'oracle choisit une valeur x aléatoire dans E avec la probabilité uniforme. Soit $b = p(x)$.
2. L'oracle calcule $y = f(x)$ et transmet la valeur de $y \in F$ à l'adversaire.
3. Au bout d'un temps polynomial, l'adversaire propose une valeur $b^\star \in \{0, 1\}$.

L'adversaire a gagné si $b^\star = b$, dans le cas contraire, il a perdu.

La probabilité de succès de l'adversaire est $\mathrm{P}_{\text{succes}}(A) = \Pr(b^\star = b)$, son avantage est $\mathrm{Av}(A) = |2\Pr(b^\star = b) - 1|$.

Exercice 1.12. Montrer que le symbole de Jacobi $\left(\frac{y}{m}\right)$ n'est pas un prédicat difficile pour la fonction $\mathrm{RSA}_{m,e}$.

Exercice 1.13. Montrer que si l'on dispose d'un oracle qui, pour tout $x \in \mathbb{Z}/m\mathbb{Z}$ fournit le chiffre de parité de x à partir de la valeur de $y = \mathrm{RSA}_{m,e}(x)$, alors on peut inverser la fonction $\mathrm{RSA}_{m,e}$ sans connaître la factorisation de m.

Exercice 1.14. Montrer que le chiffre de parité n'est pas un prédicat difficile pour la fonction EXP sur un corps fini premier.

Exercice 1.15. Pour un élément $x \in \mathbb{Z}/p\mathbb{Z}$, on appelle chiffre de poids fort de x la quantité égale à 0 si $x < p/2$ et à 1 si $x > p/2$. Montrer que si l'on dispose d'un oracle qui pour tout z tel que $z = \mathrm{EXP}_g(n)$ fournit le chiffre de poids fort de l'exposant n, alors pour tout $y \in \mathbb{Z}/p\mathbb{Z}$, tel que $y = \mathrm{EXP}_g(\ell)$, on peut reconstituer entièrement la valeur de ℓ.

Les exercices 13 et 15 ci-dessus suggèrent que le chiffre de parité peut être considéré en pratique comme un prédicat difficile pour la fonction RSA et que le chiffre de poids fort peut être considéré en pratique comme un prédicat difficile pour la fonction EXP.

L'objet de la suite de cette section est de montrer une sorte d'équivalence, sous certaines conditions, entre la propriété pour une fonction d'être à sens unique et la propriété de disposer d'un prédicat difficile. Montrons tout d'abord l'implication dans un sens :

Proposition 6.5.

Si une fonction f est injective et a un prédicat difficile, alors f est une fonction à sens unique.

Preuve Supposons pour une contradiction que la fonction f n'est pas une fonction à sens unique, c'est-à-dire qu'il existe un adversaire polynomial A qui trouve un antécédent à $y = f(x)$ avec une probabilité de succès non négligeable. Comme f est supposée injective, l'élément x est unique.

Soit p le prédicat difficile de f. Définissons un adversaire B contre le prédicat p, qui utilise A comme sous-programme :

Algorithme 6.6. adversaire contre le prédicat p

Entrée : y tel que $y = f(x)$ avec x aléatoire.

1. Transmettre la valeur y à l'adversaire A contre la fonction à sens unique. Soit $x^\star = A(y)$ la valeur rendue par l'adversaire A.
2. Si $f(x^\star) = y$, c'est-à-dire si A a réussi, poser $b^\star = p(x^\star)$.
3. Sinon, poser $b^\star$ égal à une valeur aléatoire.

Sortie : renvoyer la valeur $b^\star$.

Évaluons l'avantage de l'algorithme B. Posons $\mu(n)$ la fonction non négligeable égale à la probabilité

de succès de A. Posons E l'événement égal au succès de l'algorithme A, c'est-à-dire $f(x^\star) = y$. Dans ce cas, l'antécédent $x^\star$ étant unique, l'algorithme B a une probabilité de succès égale à 1. Dans le cas contraire, comme l'algorithme B renvoie une valeur aléatoire, sa probabilité de succès vaut 1/2. Posons $p(x) = b$.

$$\begin{aligned}\Pr(b^\star = b) &= \underbrace{\Pr(b^\star = b \mid E))}_{=1} \times \underbrace{\Pr(E)}_{\mu(n)} + \underbrace{\Pr(b^\star = b \mid \overline{E}))}_{1/2} \times \underbrace{\Pr(\overline{E})}_{1-\mu(n)} \\ &= \mu(n) + \frac{1}{2}(1-\mu(n)) \\ &= \frac{1}{2}\mu(n) + \frac{1}{2}\end{aligned}$$

L'avantage de B est donc $\mathrm{Av}(B) = 2\Pr(b^\star = b) - 1 = \mu(n)$. Il est non négligeable, ce qui contredit l'hypothèse selon laquelle p est un prédicat difficile. □

Comme le montre l'exercice suivant, le résultat de la proposition 5 ci-dessus est faux si on ne suppose pas f injective.

Exercice 1.16. Trouver une fonction f qui n'est pas à sens unique mais qui a un prédicat difficile.

Contrairement aux fonctions à sens unique, dont l'existence reste aujourd'hui une conjecture, on sait construire des fonctions qui ont un prédicat difficile. Mais ces fonctions ont un intérêt pratique limité.

7 Théorème de Goldreich-Levin

Le théorème de Goldreich-Levin est résultat fondamental qui constitue en quelque sorte une réciproque de la proposition 5. Il énonce qu'à partir d'une fonction à sens unique, on peut construire une autre fonction à sens unique qui lui est très proche et qui dispose, elle, d'un prédicat difficile explicite.

Nous en déduirons abusivement que l'existence d'un prédicat difficile caractérise les fonctions à sens unique.

Théorème 7.1. Goldreich-Levin

Soit $f : \{0,1\}^\star \to \{0,1\}^\star$ une fonction qu'on suppose être à sens unique. Soit g la fonction définie par :

$$g : \begin{array}{ccc}\{0,1\}^\star \times \{0,1\}^\star & \to & \{0,1\}^\star \times \{0,1\}^\star \\ (x,r) & \mapsto & (f(x),r)\end{array}.$$

Les deux énoncés suivants sont satisfaits :

1. La fonction g est à sens unique ;
2. La fonction booléenne $(x,r) \mapsto x \cdot r$ est un prédicat difficile pour g.

Preuve Prouvons tout d'abord le premier point et montrons que la fonction g est à sens unique.

Supposons, en vue d'une contradiction, que tel n'est pas le cas. Il existe donc un algorithme polynomial A qui sait inverser la fonction g avec une probabilité de succès non négligeable. Déduisons-en l'existence d'un algorithme B efficace qui inverse la fonction f avec une probabilité de succès non négligeable. Pour cela exhibons un tel algorithme qui utilise A comme sous-programme et évaluons ses performances. L'existence de l'algorithme B est contradictoire avec l'hypothèse selon laquelle la fonction f est à sens unique.

Algorithme 7.2. qui inverse f à partir d'un algorithme A qui inverse g

Entrée : y fourni par l'oracle qui a calculé $y = f(x)$ avec x aléatoire.

1. choisir r au hasard dans $\{0,1\}^n$;
2. Soumettre le couple (y, r) à l'algorithme A, soit $(x^\star, r^\star)$ sa réponse ;

Sortie : renvoyer la valeur $x^\star$.

Le succès de A est l'événement $r^\star = r$ et $f(x^\star) = y$. Il implique le succès de B qui est l'événement $f(x^\star) = y$. On a donc l'inégalité :

$$\Pr_{\text{succes}}(B) \geq \Pr_{\text{succes}}(A)$$

La probabilité de succès de A étant supposée non négligeable, il résulte que celle de B l'est aussi, ce qui contredit l'hypothèse selon laquelle la fonction f est à sens unique.

Montrons maintenant le deuxième point, c'est-à-dire que la fonction booléenne $(x, r) \mapsto x \cdot r$ est bien un prédicat difficile pour la fonction g.

Supposons en vue d'une contradiction qu'il existe un algorithme A qui trouve $x \cdot r$ à partir de la donnée de (y, r), où $y = f(x)$ avec x aléatoire dans $\{0,1\}^n$ et r aléatoire dans $\{0,1\}^n$. Construisons un algorithme B qui inverse la fonction f avec une probabilité de succès non négligeable. Cela montrera que, si $x \cdot r$ n'est pas un prédicat difficile, alors la fonction f n'est pas à sens unique, ce qui est contraire à l'hypothèse.

Une solution simple pour l'algorithme B, qui convient lorsque l'algorithme A donne des valeurs certaines, consiste à générer n vecteurs aléatoires r_i linéairement indépendants. Les données des $x \cdot r_i$ rendues par A permettent alors de retrouver la valeur du vecteur x en résolvant un système d'équations linéaires.

Mais cette méthode ne fonctionne pas si l'algorithme A donne des résultats incertains. Si l'avantage de A vaut par exemple la fonction non négligeable $1/n$, sa probabilité de succès vaut $\frac{1}{2} + \frac{1}{2n}$. Le succès de l'algorithme d'inversion exige que toutes les composantes soient correctes. S'il subsiste, ne serait-ce qu'une seule composante incorrecte, l'algorithme ne peut résoudre correctement le système. La probabilité de succès d'un tel algorithme est la probabilité de recevoir de l'algorithme A toutes ses réponses correctes. Or cet événement ne survient qu'avec une probabilité égale à :

$$\left(\frac{1}{2} + \frac{1}{2n}\right)^n = \frac{1}{2^n} \underbrace{\left(1 + \frac{1}{n}\right)^n}_{\text{converge vers } e \text{ quand } n \to +\infty}.$$

Il est bien connu que ce second membre est équivalent à $\frac{e}{2^n}$ au voisinage de $+\infty$. Cela montre que la probabilité de succès de l'algorithme B est négligeable. Cet algorithme ne convient donc pas pour prouver le résultat.

Il faut donc un algorithme plus subtil qui supporte de recevoir de l'algorithme A des résultats erronés, pourvu que son avantage reste non négligeable, tout en ayant une probabilité non négligeable de succès et une complexité polynomiale.

Considérons l'algorithme B suivant, destiné à inverser la fonction f. Comme le précédent, il utilise l'adversaire A comme sous-programme. Mais pour accepter des réponses erronées, il fait de nombreux appels à l'algorithme A, traite ses résultats par vote majoritaire et effectue une partie de recherche exhaustive.

Dans la définition de cet algorithme, l'entier ℓ est un paramètre qu'il s'agira de déterminer pour que la complexité de B reste polynomiale et que sa probabilité de succès soit non négligeable.

Algorithme 7.3. B **qui inverse** f **en utilisant une estimation de** $x \cdot r$

Entrée : un élément y calculé comme $y = f(x)$ avec x aléatoire dans $\{0,1\}^n$.
Soit x la valeur présumée d'un antécédent de y.

1. Choisir ℓ vecteurs indépendants $r_1, \ldots, r_\ell$ dans $\{0,1\}^n$ et attribuer ℓ valeurs arbitraires dans $\{0,1\}$ aux valeurs supposées des produits scalaires $x \cdot r_1, \ldots, x \cdot r_\ell$.
2. Pour tous les sous-ensembles d'indices I, inclus dans l'ensemble $\{1, \ldots, \ell\}$, poser $r_I = \sum_{i \in I} r_i$.
   Pour tous les indices $i \in \{1, \ldots, \ell\}$, appeler l'algorithme A avec l'entrée $(y, r_I + e_i)$, où e_i est le i-ième vecteur de la base canonique de l'espace vectoriel $\{0,1\}^n$.
   Soit $z_{i,I}$ la réponse de l'algorithme A, qui est donc une estimation de $x \cdot (r_I + e_i)$.
3. Estimer la composante $x_i^\star$ de la façon suivante :
   — Poser $x_{i,I}^\star = x \cdot r_I + z_{i,I} = x \cdot r_I + x \cdot (r_I + e_i)$, qui est une estimation de x_i sur la base de la réponse de A pour l'ensemble d'indice I.
   — Définir $x_i^\star$ par vote majoritaire :

$$\begin{cases} x_i^\star = 0 & \text{si} \quad \sum_{I \neq \emptyset} x_{i,I}^\star < 2^{\ell-1} \\ x_i^\star = 1 & \text{si} \quad \sum_{I \neq \emptyset} x_{i,I}^\star \geq 2^{\ell-1} \end{cases}$$

Sortie : Retourner le vecteur des estimations $x^\star = (x_1^\star, \ldots x_n^\star)$.

Il reste à trouver la valeur du paramètre ℓ pour que la complexité de B soit polynomiale et pour qu'il inverse bien la fonction f avec une probabilité de succès non négligeable. Pour cela, les tâches à effectuer sont les suivantes :

1. Évaluer l'avantage de A. Ici, l'algorithme A n'est pas appelé dans ses conditions normales d'utilisation. Son entrée devrait être un couple (y, r) où $y = f(x)$ avec x et r aléatoires. Or, ici, le paramètre y est constant et le paramètre r est la somme d'un vecteur de la base canonique et d'un vecteur qui appartient à un sous-espace de dimension ℓ. Le jeu n'est pas honnête pour A et il faut évaluer son avantage dans ces circonstances particulières.
2. Évaluer la probabilité de succès de l'algorithme B lorsque les valeurs arbitraires choisies pour les $x \cdot r_i$ sont exactes.
3. Trouver la valeur convenable ℓ pour que l'algorithme B reste d'une complexité polynomiale avec une probabilité de succès non négligeable.

Les deux premiers points ci-dessus sont l'objet des deux lemmes suivants :

Lemme 7.4. Succès de l'algorithme A **sur une entrée** y **constante**

Soit $\mu(n)$ l'avantage de l'algorithme A lors de son utilisation normale. Pour toute valeur y fixée, on note E_y l'événement « *l'algorithme A avec l'entrée (y, r) réussit avec une probabilité supérieure à* $1/2 + \mu(n)/4$ ».
La probabilité de l'événement E_y est supérieure ou égale à $\mu(n)/4$.

Lemme 7.5. Succès de l'algrithme B

Si les valeurs arbitraires choisies par l'algorithme B sont exactes, si l'algorithme A réussit avec une probabilité supérieure ou égale à $1/2 + \mu(n)/4$, et si $2^\ell > n/\mu(n)^2$, alors pour n assez grand, la probabilité de succès de l'algorithme B est supérieure à $1/e^4$.

Terminons tout d'abord la preuve du théorème de Goldreich-Levin en admettant provisoirement les résultats de ces deux lemmes.

Choisir l'entier ℓ minimal tel que $2^\ell > n/\mu(n)^2$, c'est-à-dire :

$$\frac{n}{\mu(n)^2} < 2^\ell \leq \frac{2n}{\mu(n)^2}.$$

Évaluons la probabilité de succès de l'algorithme B. Pour cela, considérons les trois événements suivants :

E_y « *L'algorithme A réussit sur l'entrée (y, r) avec une probabilité de succès supérieure ou égale à $1/2 + \mu(n)/4$.* »

C « *Les valeurs arbitraires choisies par l'algorithme B sont correctes.* »

S « *L'algorithme B réussit.* »

Il s'agit d'évaluer la probabilité de succès de B, égale à $\Pr(S)$. On a :

$$\Pr(S) \geq \Pr(S \wedge C \wedge E_y) = \Pr(S \mid C \wedge E_y) \times \Pr(C \wedge E_y).$$

Les événements C et E_y sont indépendants, car les valeurs arbitraires choisies par l'algorithme B sont aléatoires. Donc :

$$\Pr(S) \geq \Pr(S \mid C \wedge E_y) \times \Pr(C) \times \Pr(E_y).$$

Évaluons chaque facteur :

— D'après le lemme 5, la probabilité $\Pr(S \mid C \wedge E_y)$ est supérieure à $1/e^4$ au voisinage de l'infini.

— Le probabilité $\Pr(C)$ vaut $1/2^\ell$ qui est supérieure à $\mu(n)^2/2n$ en raison du choix de la valeur de ℓ.

— D'après le lemme 4, la probabilité $\Pr(E_y)$ est supérieure à $\mu(n)/4$.

En compilant tous ces résultats, on obtient :

$$\Pr(S) \geq \frac{\mu(n)^3}{8ne},$$

qui est non négligeable.

Il reste à évaluer la complexité de l'algorithme B et de vérifier qu'elle reste bien polynomiale en n.

Le nombre de précalculs des valeurs arbitraires vérifie, en raison du choix de l'entier ℓ :

$$2^\ell \leq \frac{2n}{\mu(n)^2}$$

Le nombre d'appels à l'algorithme A est $(2^\ell - 1) \times n$: un pour chaque sous-ensemble I non vide de $\{1, \ldots, \ell\}$ et pour chaque indice $i \in \{1, \ldots, n\}$. La complexité du calcul des estimations $x_i^\star$ est lui aussi égal à $(2^\ell - 1) \times n$ qui est inférieur à $2^\ell \times n$, donc inférieur également à $2n^2/\mu(n)^2$. La complexité totale est donc inférieure à :

$$\frac{1}{\mu(n)^2}(2n + 2n^2),$$

ce qui reste bornée par un polynôme en n. □

Il reste à établir la preuve des deux lemmes.

Preuve du lemme 4 Supposons pour une contradiction que l'événement E_y a une probabilité strictement inférieure à $\mu(n)/4$. Soit S_A le succès de A et exprimons sa probabilité selon que l'événement E_y est ou n'est pas survenu. Par définition de cet événement, lorsqu'il n'est pas satisfait, la probabilité de succès de l'algorithme A est strictement majoré par $1/2 + \mu(n)/4$. D'où :

$$\Pr(S_A) = \underbrace{\Pr(S_A \mid E_y)}_{\leq 1} \times \underbrace{\Pr(E_y)}_{< \frac{\mu(n)}{4}} + \underbrace{\Pr(S_A \mid \overline{E}_y)}_{< \frac{1}{2} + \frac{\mu(n)}{4}} \times \underbrace{\Pr(\overline{E}_y)}_{\leq 1}$$

Il en résulte que :

$$\Pr(S_A) < \frac{1}{2} + \frac{\mu(n)}{2},$$

ce qui confère à l'algorithme A un avantage strictement inférieur à $\mu(n)$, ce qui est contraire à l'hypothèse selon laquelle il est égal à $\mu(n)$. □

Preuve du lemme 5 L'algorithme B réussit si chaque estimation $x_i^\star$ est correcte, c'est-à-dire si une majorité de $x_{i,I}^\star$ est correcte. Pour tout indice i appartenant à $\{1,\ldots,n\}$ et tout sous-ensemble I non vide de $\{1,\ldots\ell\}$, on note $X_{i,I}$ la variable aléatoire égale à 0 si l'algorithme A fournit une réponse correcte lors de l'appel avec l'entrée $(y, r_I + e_i)$ et égale à 1 dans le cas contraire. En d'autres termes, on a $X_{i,I} = x_{i,I}^\star \oplus x_i$. Par hypothèse, $\Pr(x_i \neq x_{i,I}^\star) = \Pr(X_{i,I} = 1) < 1/2 - \mu(n)/4$.

Fixons l'indice $i \in \{0,\ldots,n\}$. Notons X_i le nombre de réponses fausses de l'algorithme A lors des $2^\ell - 1$ appels que fait l'algorithme B pour estimer x_i, c'est-à-dire $X_i = \sum_{I\neq\emptyset} X_{i,I}$. Le nombre moyen d'évaluations incorrectes de l'algorithme A est :

$$E(X_i) = \sum_{I\neq\emptyset} E(X_{i,I}) < 2^\ell \left(\frac{1}{2} - \frac{\mu(n)}{4}\right).$$

Donc

$$X_i - E(X_i) > X_i - 2^{\ell-1} + 2^{\ell-2}\mu(n).$$

En raison de l'estimation de x_i par vote majoritaire, l'échec de B sur l'évaluation de x_i signifie que le nombre de résultats incorrects de l'algorithme A est supérieur à $2^{\ell-1}$. En raison de l'inégalité ci-dessus, l'implication suivante est satisfaite :

$$X_i \geq 2^{\ell-1} \Rightarrow X_i - E(X_i) > 2^{\ell-2}\mu(n) \Rightarrow |Xi\_E(X_i)| > 2^{\ell-2}\mu(n).$$

Par conséquent :

$$\Pr(x_i^\star \neq x_i) = \Pr(X_i \geq 2^{\ell-1}) \leq \Pr\Big(|X_i - E(X_i)| > 2^{\ell-2}\mu(n)\Big).$$

Rappelons que l'inégalité de Bienaymé-Tchébychev énonce que :

$$\Pr(|X - E(X)| \geq \alpha) \leq \frac{\mathrm{Var}(X)}{\alpha^2}.$$

Ceci permet d'établir la majoration suivante :

$$\Pr(x_i^\star \neq x_i) \leq \frac{\mathrm{Var}(X_i)}{2^{2\ell-4}\mu(n)^2}.$$

Évaluons la variance de X_i. Si I et I' sont deux sous-ensembles différents de $\{1,\ldots,\ell\}$, c'est qu'il existe un élément i qui est dans l'un, mais qui n'est pas dans l'autre. Comme les vecteurs r_i sont choisis aléatoirement et indépendamment, les vecteurs r_I et $r_{I'}$ sont indépendants, et donc aussi les réponses de l'algorithme A sur ces valeurs. Il en résulte que les variables aléatoires $X_{i,I}$ et $X_{i,I'}$ sont indépendantes. La variance de X_i est donc la somme des variances des $X_{i,I}$:

$$\mathrm{Var}(X_i) = \sum_{I\neq\emptyset} \mathrm{Var}(X_{i,I})$$

Or $\mathrm{Var}(X_{i,I}) = E(X_{i,I}^2) - E(X_{i,I})^2$. La variable aléatoire $X_{i,I}$ prenant ses valeurs dans l'ensemble $\{0,1\}$, elle est égale à son carré, d'où $\mathrm{Var}(X_{i,I}) = E(X_{i,I}) - E(X_{i,I})^2 = E(X_{i,I})(1 - E(X_{i,I}))$. L'espérance $E(X_{i,I})$ prend ses valeurs sur l'intervalle $[0,1]$. La fonction réelle $x \mapsto x(1-x)$ admet un maximum sur cet intervalle égal à $1/4$ pour $x = 1/2$, donc $\mathrm{Var}(X_{i,I}) \leq 1/4$. Il en résulte que : $\mathrm{Var}(X_i) \leq 2^{\ell-2}$. Finalement :

$$\Pr(x_i^\star \neq x_i) \leq \frac{2^{\ell-2}}{2^{2\ell-4}\mu(n)^2} = \frac{4}{2^\ell\mu(n)^2}.$$

Si l'entier ℓ satisfait l'inégalité $2^\ell > n/\mu(n)^2$, alors :

$$\Pr(x_i^\star \neq x_i) < \frac{4}{n}.$$

Cela signifie que l'algorithme B évalue correctement chaque composante x_i avec une probabilité supérieure à $1 - 1/n$. Comme le succès de l'algorithme B est son succès pour évaluer les n composantes de x, il en résulte que :

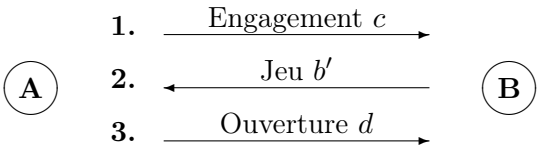
$$\Pr(x^\star = x) \geq \left(1 - \frac{4}{n}\right)^n.$$

Il est bien connu que le second membre ci-dessus tend vers $1/e^4$ quand n tend vers l'infini. La probabilité de succès de l'algorithme B est donc supérieure à cette constante pour n assez grand. □

8 Application : système de mise en gage

Une application du prédicat difficile d'une fonction à sens unique est la mise en gage. Il s'agit d'un protocole qui consiste à engager une information secrète sans la divulguer et sans pouvoir revenir dessus. Une illustration typique de la mise en gage est le jeu de pile ou face par téléphone, entre deux joueurs qui ne se font pas confiance. Ils doivent convenir d'un procédé pour empêcher la triche.

Le protocole de mise en gage comprend trois échanges.



1. Le joueur **A** choisit un symbole binaire b, égal à 0 ou à 1, et l'engage par le calcul de deux valeurs :
   — une valeur d'engagement (*commit*) c, qu'il transmet au joueur B et sur laquelle il ne pourra pas revenir.
   — une valeur d'ouverture d qu'il conserve pour plus tard.
2. Le joueur **B** tente de deviner la valeur b choisie par A et joue une valeur binaire b' qu'il transmet au joueur **A**.
3. Le joueur **A** divulgue alors la valeur d'ouverture d qui va révéler son choix b au joueur **B**.

La mise en œuvre de ce jeu requiert deux algorithmes :
— Un algorithme d'engagement ENG, qui à partir du symbole binaire b produit un couple (c, d) d'engagement et d'ouverture.
— Une algorithme d'ouverture OUV, qui à partir du couple (c, d) d'engagement et d'ouverture, vérifie l'absence de triche et révèle la valeur du symbole binaire : $\mathrm{OUV}(c, d) = b$.

Pour que le jeu fonctionnne, ces deux fonctions doivent satisfaire :

$$\forall b \in \{0, 1\},\ \mathrm{OUV} \circ \mathrm{ENG}(b) = b.$$

Mais pour empêcher la triche, les deux propriétés de sécurité suivantes sont requises :
— La propriété de dissimulation (*hidding*) : la connaissance de la valeur d'engagement c ne doit rien révéler sur la valeur binaire b choisie par le joueur **A**. En d'autres termes, il ne doit pas exister d'algorithme efficace qui permet de trouver b à partir de c.
— Le propriété d'enchérissement (*biding*) : une fois que c a été révélé, le joueur **A** ne peut pas revenir sur la valeur binaire choisie b qui sera révélée par la valeur d'ouverture d. En d'autres termes, il ne doit pas être possible de trouver deux valeurs d'ouverture d et d' telles que $\mathrm{OUV}(c, d) \neq \mathrm{OUV}(c, d')$. Le joueur **A** ne doit pas pouvoir modifier son choix après la révélation de la valeur b' choisie par le joueur **B**.

Exercice 1.17. Proposer un système de mise en gage pratique à l'aide d'une fonction à sens unique injective f et d'un prédicat supposé difficile de f.

2
Génération de pseudo-aléa

Dans un chiffrement à flot (*stream cipher*), le message en clair est représenté sous forme d'une suite de symboles binaires sur l'alphabet $\{0,1\}$. Chaque symbole est combiné par un *ou exclusif* à un symbole issu d'une suite pseudo-aléatoire. Cette opération masque l'information portée par le message. À la réception, le cryptogramme ainsi formé est combiné à la même séquence pseudo-aléatoire par la même opération de *ou exclusif*, ce qui permet le reconstituer le message initial.

Les propriétés attendues d'un générateur de pseudo-aléa sont les suivantes :

— La séquence soit être reproductible à l'identique par l'émetteur et le destinataire à partir d'un germe partagé ;

— Elle doit présenter toutes les apparence d'une véritable séquence aléatoire : être imprévisible et indiscernable d'un tirage par jeu de pile ou face.

Une suite binaire qui satisfait ces deux propriétés s'appelle une *suite pseudo-aléatoire*. Elle doit être assez longue pour chiffrer de longs messages, mais elle doit être produite à partir d'un germe assez court pour qu'il puisse être échangé de manière commode par les deux protagonistes. Un algorithme qui produit de telles suites s'appelle un *générateur de pseudo-aléa*. Intuitivement, un tel générateur amplifie l'aléa du germe. Mais la théorie de l'information nous apprend que l'entropie d'une séquence aléatoire ne peut en aucun cas résulter d'un calcul. Elle ne peut en aucun cas être supérieure à l'entropie du germe qui la produit. Le caractère pseudo-aléatoire est par conséquent une notion calculatoire qu'il s'agit de définir précisément. Le principal résultat présenté dans ce chapitre est l'équivalence entre l'existence de fonctions à sens unique et celle des générateurs de pseudo-aléa.

1 Indistinguabilité des variables aléatoires

La qualité d'un générateur de pseudo-aléa (GPA) se mesure à la difficulté de faire la différence entre l'observation de valeurs issues de celui-ci et celles issues d'un véritable générateur d'aléa (GDA). Cette qualité se quantifie par la chance de gagner au jeu de l'aléa.

Ce jeu fait intervenir deux joueurs : un oracle qui fournit des réalisations choisies parmi deux variables aléatoires, et un adversaire, appelé *distingueur*[1], chargé de reconnaître de quelle variable sont issues les réalisations qui lui sont présentées.

Jeu 1.1. de l'aléa

Les partenaires disposent de la connaissance de deux variables aléatoires X et Y et de leurs lois de probabilité.

1. L'oracle choisit un symbole binaire aléatoire $b \in_{\mathrm{R}} \{0,1\}$ avec la probabilité uniforme.
2. Si $b = 0$, alors l'oracle envoie au distingueur des réalisations de la variable X,
3. Si $b = 1$, alors l'oracle envoie au distingueur des réalisations de la variable Y.
4. Au bout d'un certain nombre de réalisations, le distingueur déclare avoir reconnu de quelle variable aléatoire les réalisations transmises sont issues. S'il a reconnu des réalisations de X, il renvoie $b^\star = 0$ et s'il a reconnu des réalisations de Y, il renvoie $b^\star = 1$.
5. Le distingueur a gagné si $b^\star = b$. Il a perdu dans le cas contraire.

Le succès du distingueur est l'événement $\{b = b^\star\}$. Sa probabilité de succès est $\Pr(b = b^\star)$.

1. Ce néologisme est une francisation du terme anglais *distinguisher*

Le jeu de l'aléa fait partie d'une famille de jeux dans lequel un oracle choisit un symbole binaire b dans l'ensemble $\{0,1\}$ avec la probabilité uniforme, et connu du seul oracle. Un distingueur est de manière générale un algorithme chargé de trouver avec quelle valeur de b l'oracle opère. Il renvoie une estimation $b^\star$ de b.

La proposition suivante donne une expression générale de l'avantage d'un distingueur lorsque la valeur b choisie par l'oracle suit une loi uniforme, chaque valeur étant tirée indépendamment des autres avec une probabilité égale à $1/2$.

Proposition 1.2. avantage d'un distingueur lorsque le choix est binaire et équiprobable

Lorsque la valeur de b est aléatoire et équidistribuée, l'avantage d'un distingueur D qui renvoie l'estimation $b^\star$ de b a pour expression :

$$\mathrm{Av}(D) = \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1)\Big| \tag{2.1}$$

Preuve La probabilité de succès du distingueur D est $\mathrm{P}_{\text{succes}}(D) = \Pr(b = b^\star)$. Cette expression se décompose selon la valeur de $b^\star$ en :

$$\begin{aligned}
\mathrm{P}_{\text{succes}}(D) &= \Pr(b = 0 \wedge b^\star = 0) + \Pr(b = 1 \wedge b^\star = 1) \\
&= \Pr(b^\star = 0 \mid b = 0) \times \underbrace{\Pr(b = 0)}_{= 1/2} + \Pr(b^\star = 1 \mid b = 1) \times \underbrace{\Pr(b = 1)}_{= 1/2} \\
&= \frac{1}{2}\Big(\Pr(b^\star = 0 \mid b = 0) + \Pr(b^\star = 1 \mid b = 1)\Big)
\end{aligned}$$

L'expression de l'avantage est $\mathrm{Av}(D) = |2\mathrm{P}_{\text{succes}}(D) - 1|$. En utilisant l'expression de la probabilité de succès ci-dessus, et en remarquant que $1 = \Pr(b^\star = 0 \mid b = 1) + \Pr(b^\star = 1 \mid b = 1)$, on a :

$$\begin{aligned}
\mathrm{Av}(D) &= \Big|\Pr(b^\star = 0 \mid b = 0) + \Pr(b^\star = 1 \mid b = 1) - \Pr(b^\star = 1 \mid b = 1) - \Pr(b^\star = 0 \mid b = 1)\Big| \\
&= \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1)\Big|
\end{aligned}$$

□

L'avantage d'un distingueur D pour discerner entre les variables aléatoires X et Y représente un éloignement qu'il observe entre ces deux variables. Cette quantité se note $\mathrm{Av}_{X,Y}(D)$.

Comme $\mathrm{Av}_{XY}(D)$ est un réel compris entre 0 et 1, l'ensemble de tous les avantages est une partie non vide et majorée de $\mathbb{R}$, ce qui donne un sens à la définition suivante :

Définition 1.3. distance calculatoire

Soient X et Y deux variables aléatoires. La *distance calculatoire* entre X et Y est la borne supérieure de l'ensemble des avantages de tous les distingueurs chargés de distinguer entre X et Y :

$$\Delta(X,Y) = \sup_{D \in \mathcal{D}} \mathrm{Av}_{X,Y}(D),$$

où $\mathcal{D}$ désigne l'ensemble de tous les distingueurs de X et Y.

La distance caculatoire entre X et Y si l'avantage de tout distingueur est nul, ce qui signifie que les variables X et Y sont impossible à distinguer, et elle vaut 1 s'il existe un distingueur qui sait distinguer l'une de l'autre avec certitude. Cela représente bien une distance et cette appellation n'a rien de fortuit. Cette quantité satisfait en effet tous les axiomes d'une distance au sens métrique du terme comme l'énonce la proposition qui suit :

Proposition 1.4. la distance calculatoire est une distance

La distance calculatoire satisfait les trois axiomes d'une distance. Soient X, Y et Z trois variables aléatoires :

A1 séparabilité : $\Delta(X,Y) = 0 \Leftrightarrow X = Y$,

A2 symétrie : $\Delta(X,Y) = \Delta(Y,X)$,

A3 inégalité triangulaire :

$$\Delta(X,Z) \leq \Delta(X,Y) + \Delta(Y,Z). \tag{2.2}$$

Preuve

Séparabilité : Si les variables aléatoires X et Y ont la même loi, la réponse du distingueur est indépendante de la loi qui a été choisie par l'oracle. L'expression de l'avantage de D donnée par la relation 2.1 montre que celui-ci est nul.

Réciproquement, supposons les variables X et Y de loi différente, et montrons l'existence d'un distingueur d'avantage strictement positif.

Soit E l'ensemble des valeurs prises par les variables aléatoires X et Y. Considérons les trois ensembles suivants :

— l'ensemble des valeurs plus probablement prises par X que par Y :
$$U = \{x \in E \mid \Pr(X = x) > \Pr(Y = x)\},$$
— l'ensemble des valeur prises par X et par Y avec la même probabilité :
$$V = \{x \in E \mid \Pr(X = x) = \Pr(Y = x)\} \text{ et}$$
— l'ensemble des valeurs plus probablement prises par Y que par X.
$$W = \{x \in E \mid \Pr(X = x) < \Pr(Y = x)\}.$$

Si les deux variables aléatoires X et Y sont de loi différentes, alors les ensembles U et W sont non vides. Considérons alors l'adversaire A suivant :

Algorithme 1.5.

Entrée : une réalisation x de la variable X ou de la variable Y.

1. si $x \in U$, renvoyer $b^\star = 0$ qui signifie que x est probablement une réalisation de X.
2. si $x \in W$, renvoyer $b^\star = 1$ qui signifie que x est probablement une réalisation de Y.
3. sinon, c'est-à-dire si $x \in V$ renvoyer une valeur $b^\star$ aléatoire.

Montrons que l'avantage de cet adversaire est strictement positif. Son expression est :

$$\mathrm{Av}(A) = \left|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1)\right|$$

L'algorithme A renvoie $b^\star = 0$ dans deux cas seulement : l'entrée x appartient à l'ensemble U, et l'entrée x appartient à l'ensemble V et la valeur tirée au hasard est $b^\star = 0$. Par conséquent :

$$\mathrm{Av}(A) = \Big|\underbrace{\Pr(x \in U \mid b = 0)}_{= \Pr(X \in U)} + \underbrace{\frac{1}{2}\Pr(x \in V \mid b = 0)}_{(1)} - \underbrace{\Pr(x \in U \mid b = 1)}_{= \Pr(Y \in U)} - \underbrace{\frac{1}{2}\Pr(x \in V \mid b = 1)}_{(2)}\Big|$$

Lorsque $x \in V$ il est autant probable qu'il soit une réalisation de X que de Y, les termes (1) et (2) s'annulent donc. Par définition de l'ensemble U, la différence des autres termes est strictement positive. L'avantage de l'algorithme A est donc strictement positif.

Symétrie : Pour un adversaire A au jeu de l'aléa, on a $\mathrm{Av}_{XY}(A) = \mathrm{Av}_{YX}(A)$ en raison de la valeur absolue qui définit l'avantage. Cette égalité pour tout adversaire A implique une égalité semblable pour la borne supérieure sur tous les adversaires.

Inégalité triangulaire : Soit A un adversaire au jeu de l'aléa. On note U la variable aléatoire dont il reçoit les réalisations. S'il doit distinguer entre X et Z, on a $U = X$ si $b = 0$ et $U = Z$ si $b = 1$. Soit $b^\star$ la

réponse de A. L'expression de l'avantage donné par la relation 2.1 de la proposition 2 s'écrit :

$$\mathrm{Av}_{XZ}(A) = \Big|\Pr(b^\star = 0 \mid U = X) - \Pr(b^\star = 0 \mid U = Z)\Big|.$$

Introduisons un terme intermédiaire et appliquons l'inégalité triangulaire de la valeur absolue :

$$\begin{aligned}\mathrm{Av}_{XZ}(A) &= \Big|\Pr(b^\star = 0 \mid U = X) - \Pr(b^\star = 0 \mid U = Y) + \Pr(b^\star = 0 \mid U = Y) - \Pr(b^\star = 0 \mid U = Z)\Big| \\ &\leq \underbrace{\Big|\Pr(b^\star = 0 \mid U = X) - \Pr(b^\star = 0 \mid U = Y)\Big|}_{\mathrm{Av}_{XY}(A)} + \underbrace{\Big|\Pr(b^\star = 0 \mid U = Y - \Pr(b^\star = 0 \mid U = Z)\Big|}_{\mathrm{Av}_{YZ}(A)}\end{aligned}$$

Cette inégalité est satisfaite pour tout adversaire A au jeu de l'aléa. L'inégalité triangulaire pour la distance calculatoire s'en déduit. Soit ε un réel strictement positif quelconque. Il existe un distingueur A_ε tel que la distance calculatoire $\Delta(X, Y)$ satisfait :

$$\Delta(X, Z) - \varepsilon \leq \mathrm{Av}_{XZ}(A_\varepsilon) \leq \mathrm{Av}_{XY}(A_\varepsilon) + \mathrm{Av}_{YZ}(A_\varepsilon) \leq \Delta(X, Y) + \Delta(Y, Z).$$

Comme cette inégalité est satisfaite pour tout ε, l'inégalité triangulaire pour la distance calculatoire (2.2) en résulte par passage à la limite en faisant tendre ε vers 0. □

Exercice 2.1. Pour deux variables aléatoires X et Y, on note XY la variable aléatoire dont les réalisations sont celles du couple (X, Y). Soient X, Y, et Z trois variables aléatoires, X et Y étant à valeurs sur le même ensemble. Démontrer que :

$$\Delta(XZ, YZ) = \Delta(X, Y).$$

Le résultat pratique de cet exercice est une règle de simplification de la distance calculatoire de variables aléatoires lorsqu'elles sont le résultat de la concaténation à droite (ou à gauche) d'une même variable aléatoire Z.

2 Variables aléatoires calculatoirement indistinguables

Deux variables dont dites calculatoirement indistinguables s'il est pratiquement impossible de faire la différence entre des réalisations de l'une et des réalisations de l'autre. L'impossibilité pratique exprime qu'on se limite à des distingueurs de complexité polynomiale et que l'avantage de ceux-ci ne peuvent qu'être négligeables. Ces notions étant asymptotiques, l'indistinguabilité calculatoire s'applique, non pas à un couple de variables aléatoires donné, mais un un couple de familles de variables aléatoires indexées par l'ensemble $\mathbb{N}$ des entiers naturels. L'indice n agit comme un paramètre de sécurité.

Définition 2.1. Famille de variables aléatoires calculatoirement indistinguables

Soient $(X_n)_{n\in\mathbb{N}}$ et $(Y_n)_{n\in\mathbb{N}}$ deux familles de variables aléatoires binaires sur le même ensemble E_n. On dit que ces deux familles sont *calculatoirement indistinguables* si l'avantage de tout distingueur de complexité polynomiale pour distinguer X_n de Y_n au jeu de l'aléa un avantage négligeable en n.

Comme on se limite à des distingueurs dont la complexité est polynomiale, le nombre d'échantillons utilisés pour faire la distinction doit lui-même être borné par un polynôme du paramètre de sécurité.

Dans le jeu de l'aléa, le distingueur est supposé recevoir le nombre nécessaire d'échantillons de la variable aléatoire pour conclure. Le résultat surprenant ci-dessous énonce que le caractère non indistinguable de familles de variables aléatoires ne dépend pas du nombre d'échantillon reçu et qu'un distingueur peut conclure sur la base de la donnée d'un seul échantillon. La démonstration de ce théorème est aussi l'occasion de présenter une technique de démonstration puissante appelée *technique hybride*.

Théorème 2.2. distinguabilité sur la base d'une seule observation

Les familles de variables aléatoires $(X_n)_{n\in\mathbb{N}}$ et $(Y_n)_{n\in\mathbb{N}}$ sont calculatoirement distinguables si et seulement si il existe un distingueur polynomial qui distingue l'une de l'autre avec un avantage non négligeable sur la base de l'observation d'une seule réalisation de X_n ou de Y_n.

Preuve La preuve dans un sens est immédiate.

Supposons l'existence d'un distingueur polynomial qui distingue X_n de Y_n avec un avantage non négligeable sur la base d'un nombre d'observation inférieur à m, l'entier m étant majoré par un polynôme en n. Construisons un distingueur $D^\star$ qui distingue X_n de Y_n sur la base d'une seule réalisation, toujours avec une complexité polynomiale et un avantage non négligeable. Définissons tout d'abord $D^\star$ qui utilise le distingueur D comme sous-programme.

Algorithme 2.3. $D^\star$

Entrée : u, une réalisation de X_n ou de Y_n selon le choixi b de l'oracle.

1. Choisir un entier k aléatoire dans l'ensemble $\{1,\ldots,m\}$.
2. Pour cette valeur de k, construire le vecteur aléatoire hybride h dont les $k-1$ premières composantes sont des réalisations de X_n, la k-ième composante est l'entrée u et les dernières composantes sont des réalisations de Y_n :

$$h = (\underbrace{x^1,\ldots,x^{k-1}}_{\text{réalisations de } X_n},\ u,\ \underbrace{y^{k+1},\ldots,y^m}_{\text{réalisations de } Y_n}).$$

3. Soumettre les m termes qui constituent le vecteur aléatoire h à l'algorithme D.

Sortie : Renvoyer la réponse $b^\star$ rendue par D.

L'heuristique qui guide cet algorithme est la suivante : si D reconnaît m réalisations de X_n, c'est probablement que u est lui-même une réalisation de X_n, alors que s'il reconnaît m réalisations de Y_n, c'est que u est probablement une réalisation de Y_n.

L'algorithme D étant supposé polynomial, il en est de même pour $D^\star$.

Il suffit maintenant de montrer que si l'algorithme D reconnaît X_n de Y_n avec un avantage non négligeable en n, alors il en est de même pour l'algorithme $D^\star$.

Pour tout entier k compris entre 1 et m, notons H_k la variable aléatoire hybride dont les k premières composantes sont des réalisations de X_n et les $m-k$ suivantes sont des réalisations de Y_n :

$$H_k = (X^1,\ldots,X^k,Y^{k+1},\ldots,Y^m).$$

Notons H le vecteur aléatoire de dimension m égale à l'entrée du distingueur D. Remarquons que si l'entrée u est une réalisation de X_n, alors H est une réalisation de H_k :

$$(X^1,\ldots,X^{k-1},X,Y^{k+1},\ldots,Y^m) = H_k,$$

alors que si u est une réalisation de Y_n, alors H est une réalisation de H_{k-1} :

$$(X^1,\ldots,X^{k-1},Y,Y^{k+1},\ldots,Y^m) = H_{k-1}.$$

Le succès moyen de l'algorithme $D^\star$ est la moyenne des comportements pour tous les choix possibles de l'entier aléatoire k :

$$\begin{aligned}\Pr(D^\star = 0 \mid U = Y) &= \frac{1}{m}\sum_{k=1}^{m}\Pr(D = 0 \mid H = H_{k-1})\\ \Pr(D^\star = 0 \mid U = X) &= \frac{1}{m}\sum_{k=1}^{m}\Pr(D = 0 \mid H = H_k)\end{aligned}$$

L'avantage de $D^\star$ pour reconnaître X_n de Y_n est la valeur absolue de la différence des deux quantités ci-dessus :

$$\mathrm{Av}_{X_nY_n}(D^\star) = \frac{1}{m}\left|\sum_{k=1}^{m}\Pr(D=0 \mid H=H_{k-1}) - \sum_{k=1}^{m}\Pr(D=0 \mid H=H_k)\right|$$

Lors du développement de ces sommes, les termes intermédiaires s'annulent, et il ne subsiste que les termes extrêmes :

$$\mathrm{Av}_{X_nY_n}(D^\star) = \frac{1}{m}\underbrace{\left|\Pr(D=0 \mid H=H_0) - \Pr(D=0 \mid H=H_m)\right|}_{\mathrm{Av}_{H_0H_m}(D)}.$$

Remarquons simplement que les m composantes de H_0 sont toutes des réalisations de la variable aléatoire Y_n et que les m composantes de H_m sont toutes des réalisation de X_n. Le terme sur l'accolade ci-dessus n'est rien d'autre que l'avantage de D pour reconnaître X_n de Y_n :

$$\mathrm{Av}_{X_nY_n}(D^\star) = \frac{1}{m}\,\mathrm{Av}_{X_nY_n}(D^\star).$$

L'entier m étant majoré par un polynôme en n et l'avantage du distingueur D étant supposé non négligeable, il en est de même pour l'avantage du distingueur $D^\star$, et c'est ce qu'il fallait démontrer. □

Lorsqu'on supposera l'existence d'un distingueur polynomial avec avantage non négligeable pour reconnaître deux variables aléatoires, nous pourrons toujours supposer que ce distingueur donne sa réponse sur la base d'une seule observation au cours du jeu de l'aléa.

3 Générateur de pseudo aléa

Intuitivement, un générateur de pseudo aléa est un dispositif capable de produire par un calcul de longues séquences qui sont calculatoirement indistinguable d'un véritable aléa. Un tel dispositif est modélisé par une fonction g qui, à partir d'un germe court, produit une plus longue séquence de termes. La valeur produite par un générateur de pseudo aléa est une réalisation de variable aléatoire image $g(S)$, où S est la variable aléatoire qui produit le germe. Ce germe est par exemple une clé secrète de chiffrement qui se trouve être *a priori* réalisé par une variable uniforme sur l'espace des clés. Le caractère pseudo aléatoire de la séquence $g(S)$ exprime la difficulté de reconnaître les réalisations de $g(S)$ de celles d'une variable aléatoire uniforme sur l'ensemble des valeurs de la fonction g.

La notion d'indistinguabilité calculatoire étant asymptotique, un générateur aléatoire désigne une famille de fonction. Pour tout entier n, notons U_n la variable aléatoire de loi uniforme sur l'ensemble $\{0,1\}^n$.

Définition 3.1. Générateur de pseudo-aléa

Un générateur de pseudo aléa (GPA) est une famille de fonctions $(g_n)_{n\in\mathbb{N}}$, calculables par un algorithme efficace, où pour tout entier n, la fonction $g_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ a les deux propriétés suivantes :

1. Propriété d'expansion : la fonction $\ell : \mathbb{N} \to \mathbb{N}$ est strictement croissante, ce qui revient à dire que la taille de $g_n(x)$ est toujours strictement supérieure à celle de x.
2. Propriété de pseudo aléa : les familles $(g_n(U_n))_{n\in\mathbb{N}}$ et $(U_{\ell(n)})_{n\in\mathbb{N}}$ sont calculatoirement indistinguables.

En d'autres termes, dire que la famille $(g_n)_{n\in\mathbb{N}}$ est un générateur de pseudo aléa signifie que la distance calculatoire $\Delta(U_{\ell(n)}, g_n(U_n))$ est une fonction négligeable de n.

Le caractère pseudo aléatoire est une notion calculatoire. L'entropie de la variable aléatoire $g_n(U_n)$ vaut au plus n, qui est strictement inférieur à l'entropie de la variable aléatoire $U_{\ell(n)}$ qui vaut $\ell(n)$.

Si la fonction $(g_n)_{n\in\mathbb{N}}$ n'est pas un générateur de pseudo aléa, alors il existe un distingueur D qui reconnaît les réalisation de $g_n(U_n)$ des réalisations de $U_{\ell(n)}$ avec un avantage non négligeable. De plus, en raison du résultat du théorème 2, on pourra toujours supposer que ce distingueur donne sa réponse sur la base d'une seule et unique observation.

Par abus et pour éviter éviter la lourdeur dans les énoncés, on désignera sous le terme de *générateur de pseudo aléa* un élément particulier d'une famille de fonctions sur l'ensemble $\{0,1\}^n$, en sous-entendant l'indice n dans l'écriture.

Exercice 2.2. On suppose que la famille $(g_n)_{n\in\mathbb{N}}$ de fonctions $\{0,1\}^n \to \{0,1\}^m$, avec $m > n$ est un générateur de pseudo aléa. Démontrer que la famille de fonctions $(h_n)_{n\in\mathbb{N}}$ où :

$$h_n : \begin{array}{ccc} \{0,1\}^{n+k} & \to & \{0,1\}^{m+k} \\ (x,y) & \mapsto & \big(g(x),y\big) \end{array},$$

est également un générateur de pseudo aléa.

Tout comme les fonctions à sens uniques, l'existence des générateurs de pseudo aléa est aujourd'hui une conjecture qui reste ouverte. Cependant, l'exercice suivant montre que leur existence implique celle des fonctions à sens unique.

Exercice 2.3. Soit $g : \{0,1\}^n \to \{0,1\}^{2n}$ un générateur de pseudo aléa d'expansion double. Démontrer que la fonction g est à sens unique. La réciproque est-elle toujours vraie ?

Dans le jeu de l'aléa, on convient qu'une valeur $b = 0$ choisie par l'oracle correspond à la production d'un pseudo aléa, alors qu'une valeur $b = 1$ correspond à la production d'un véritable aléa.

Le distingueur D d'un générateur de pseudo-aléa n'a accès qu'à la réalisation y reçue de l'oracle, et n'a aucune connaissance de la valeur b choisie par lui. Le succès de D ne peut par conséquent reposer que sur l'appartenance ou non de son entrée y à l'image de la fonction g. Un distingueur D peut reconnaître $y \in \mathrm{Im}(g)$ comme pseudo aléatoire alors qu'il est en fait le résultat d'un tirage véritablement aléatoire. Son avantage au jeu de l'aléa ne peut donc jamais atteindre la valeur 1.

En effet, si $b^\star$ est la réponse de D, son avantage vaut :

$$\mathrm{Av}(D) = \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1)\Big|$$

Lorsque $b = 0$, alors nécessairement le défi y appartient à l'image de g. Comme D ne peut rendre son résultat que sur l'appartenance ou non de y à $\mathrm{Im}(g)$, on a :

$$\Pr(b^\star = 0 \mid b = 0) = \Pr(b^\star = 0 \mid y \in \mathrm{Im}(g)). \tag{2.3}$$

Mais lorsque $b = 1$ il se peut que le défi appartienne à l'image de g comme il se peut qu'il n'y appartienne pas, donc :

$$\Pr(b^\star = 0 \mid b = 1) = \Pr(b^\star = 0 \mid y \in \mathrm{Im}(g))\times\Pr(y \in \mathrm{Im}(g)) + \Pr(b^\star = 0 \mid y \notin \mathrm{Im}(g))\times\Pr(y \notin \mathrm{Im}(g)) \tag{2.4}$$

Par différence des équations 2.3 et 2.4,

$$\mathrm{Av}(D) = \underbrace{\Big(1 - \Pr(y \in \mathrm{Im}(g))\Big)}_{<1} \underbrace{\Big|\Pr(b^\star = 0 \mid y \in \mathrm{Im}(g)) - \Pr(b^\star = 0 \mid y \notin \mathrm{Im}(g))\Big|}_{\mathrm{Av}^\star(D) \le 1} \tag{2.5}$$

En particulier, l'avantage de D est strictement inférieur à 1. Pour cette raison, et pour mieux juger des performance d'un distingueur de GPA, il peut être préférable de considérer un jeu, appelé *jeu du pseudo aléa*, où l'oracle choisit un défi y dans l'image de g si $b = 0$ et hors de l'image de g lorsque $b = 1$. Dans ce jeu, l'avantage de D est noté $\mathrm{Av}^\star(D)$ et correspond au deuxième facteur du second membre de l'équation 2.5 Le jeu du pseudo aléa contre le générateur de pseudo aléa $g : \{0;1\}^n \to \{0,1\}^\ell$, avec $\ell > n$ oppose un oracle et un distingueur. Il est formellement défini ainsi :

Jeu 3.2. du pseudo aléa

1. L'oracle choisit un symbole binaire aléatoire $b \in_{\mathrm{R}} \{0,1\}$ avec la probabilité uniforme.
   — Si $b = 0$, alors l'oracle envoie au distingueur une donnée pseudo aléatoires y, égale à $g(x)$ où x est tirée aléatoirement dans $\{0,1\}^n$.
   — Si $b = 1$, alors l'oracle envoie au distingueur une donnée y choisie aléatoirement dans $\{0,1\}^\ell \setminus \mathrm{Im}(g)$.
2. Après un temps polynomial, le distingueur renvoie une estimation $b^\star$ de b.
3. Le distingueur a gagné si $b^\star = b$. Il a perdu dans le cas contraire.

Pour que ce jeu du pseudo aléa ait une réalité, il est nécessaire de supposer l'existence d'un algorithme efficace qui tire au hasard un élément de le complémentaire de l'image du générateur g, ce qui n'est pas toujours assuré.

Comme le permet le résultat du théorème 2, page 29 dans la définition du jeu du pseudo aléa, le distingueur ne reçoit qu'une seule réalisation – aléatoire ou pseudo aléatoire – pour construire sa réponse.

Les quantités $\mathrm{Av}(D)$ et $\mathrm{Av}^\star(D)$ ne diffèrent que par le facteur $(1 - \Pr(y \in \mathrm{Im}(g)))$ qui vaut $1-1/2^k$, où k est l'expansion du générateur. Ce facteur est appartient donc toujours à l'intervalle $[1/2, 1]$. Il en résulte que les deux formes d'avantage $\mathrm{Av}(D)$ et $\mathrm{Av}^\star(D)$ sont en même temps négligeables ou non négligeables. On pourra donc évaluer l'avantage d'un distingueur indifféremment au jeu de l'aléa ou au jeu du pseudo aléa selon ce qui est le plus commode.

Exercice 2.4. Soit g un générateur de pseudo aléa injectif d'expansion k symboles binaires $g : \{0,1\}^n \to \{0,1\}^{n+k}$. Montrer qu'un distingueur de g a un avantage inférieur ou égal à $1 - 1/2^k$ au jeu de l'aléa.

Exercice 2.5. *Concaténation de deux générateurs de pseudo-aléa*

Soient g et h deux générateurs de pseudo aléa $\{0,1\}^n \to \{0,1\}^m$. Démontrer que la fonction

$$f : \begin{array}{ccc} \{0,1\}^{2n} & \to & \{0,1\}^{2m} \\ (x,y) & \mapsto & \big(g(x), h(y)\big) \end{array},$$

est un générateur de pseudo-aléa.
Indication : utiliser la distance calculatoire.

Exercice 2.6. *Composition de générateurs de pseudo-aléa*

Soit :

$$g : \begin{array}{ccc} \{0,1\}^n & \to & \{0,1\}^{2n} \\ x & \mapsto & g(x) = \big(g_0(x), g_1(x)\big) \end{array}$$

un générateur de pseudo-aléa d'expansion double. Montrer que la fonction

$$h : \begin{array}{ccc} \{0,1\}^n & \to & \{0,1\}^{3n} \\ x & \mapsto & h(x) = \big(\, g\big(g_0(x)\big), g_1(x)\,\big) \end{array}$$

est un générateur d'aléa d'expansion triple.
Indication : Utiliser des variables aléatoires hybrides.

Exercice 2.7. *Distance calculatoire à la variable uniforme de la concaténation de générateurs de pseudo-aléa*

On suppose que $g : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ un générateur de pseudo-aléa. Pour un entier $m \geq 2$, on considère la fonction h_m définie par :

$$h_m : \begin{array}{ccc} \{0,1\}^{n\times m} & \to & \{0,1\}^{\ell(n)\times m} \\ (x_1,\ldots,x_m) & \mapsto & \big(g(x_1),\ldots,g(x_m)\big) \end{array}.$$

Montrer que la distance calculatoire de $h(U_{n\times m})$ à la variable uniforme $U_{\ell(n)\times n}$ satisfait :

$$\Delta\big(h_m(U_{n\times m}), U_{\ell(n)\times m}\big) \leq m\Delta\big(g(U_n), U_{\ell(n)}\big).$$

4 Constructions

Ce paragraphe est consacré à la construction effective d'un générateur de pseudo-aléa à partir du prédicat difficile d'une fonction à sens unique. Le chiffre de parité de la fonction RSA ou de la fonction SQR, ainsi que le chiffre de poids fort de la fonction EXP sont aujourd'hui des candidats pour mettre en œuvre de façon pratique de tels générateurs.

Par ailleurs, l'exercice 3 ci-dessus montre qu'un générateur de pseudo-aléa est aussi une fonction à sens unique. Par conséquent l'existence d'un générateur de pseudo-aléa implique l'existence de fonctions à sens unique. L'objet de cette section est de démontrer la réciproque de façon constructive, c'est-à-dire qu'un générateur de pseudo-aléa peut se construire à partir d'une fonction à sens unique. Ceci montrera finalement l'équivalence de l'existence de ces deux primitives cryptographiques.

4.1 Générateur de pseudo-aléa d'un symbole binaire d'expansion

Montrons tout d'abord que le prédicat difficile d'une fonction à sens unique permet de construire un générateur de pseudo-aléa $\{0,1\}^n \to \{0,1\}^{n+1}$ d'expansion un symbole binaire. Cette construction, certes d'un intérêt plutôt limité, constitue la première étape d'une construction plus générale utilisable en pratique.

Théorème 4.1. générateur de pseudo-aléa d'un symbole binaire d'expansion

Soit $f : \{0,1\}^n \to \{0,1\}^n$ une permutation à sens unique, et $p : \{0,1\}^n \to \{0,1\}$ un prédicat difficile pour la fonction f. Alors la fonction g définie par :

$$g : \begin{array}{ccc} \{0,1\}^n & \to & \{0,1\}^{n+1} \\ x & \mapsto & (f(x), p(x)) \end{array}$$

est un générateur de pseudo aléa d'un symbole binaire d'expansion.

Aujourd'hui, la fonction RSA et son chiffre de parité permettent de construire un tel générateur de pseudo aléa. Un autre exemple est donné par la fonction EXP et son chiffre de poids fort. La fonction carrée fournit un troisième exemple applicable (voir exercice 8 ci-après).

Preuve Supposons, en vue d'une contradiction, que la fonction g n'est pas un générateur de pseudo aléa. Il existe donc un distingueur D polynomial qui reconnaît les réalisations de $g(U_n)$ des réalisations de U_{n+1} avec un avantage non négligeable. Notons $\mu(n)$ l'avantage du distingueur D. Rappelons également qu'on peut supposer que le distingueur D donne sa réponse sur la base d'une seule observation. Montrons qu'alors le prédicat p n'est pas difficile, ce qui est contraire à l'hypothèse et achèvera la preuve. Pour cela,

exhibons un adversaire A contre le prédicat p, capable de trouver la valeur de $p(x)$ à partir de la valeur de $f(x)$ avec un avantage non négligeable.

Algorithme 4.2. *A qui trouve le prédicat difficile de la fonction f*

Entrée : y calculé comme image $f(x)$ pour un élément x de E choisi aléatoirement.

1. choisir un élément r aléatoire dans $\{0,1\}$.
2. soumettre le couple (y,r) au distingueur D, soit $d^\star \in \{0,1\}$ sa réponse.
3. si $d^\star = 0$ alors rendre $b^\star = r$, sinon rendre $b^\star = 1 - r = \overline{r}$.

L'heuristique qui guide cet algorithme est la suivante : si le couple (y,r) est reconnu comme pseudo aléatoire, et non pas purement aléatoire, c'est que r est la valeur du prédicat $p(x)$. Dans le cas contraire, c'est que r n'est pas la valeur de $p(x)$, sa valeur est donc le complément de r.

Notons que, la fonction f étant bijective, l'entrée y de l'algorithme A suit la loi uniforme. L'avantage de A est donné par :

$$\mathrm{Av}(A) = \Big|\Pr(b^\star = 0 \mid p(x) = r) - \Pr\left(b^\star = 0 \mid p(x) \neq r\right)\Big|.$$

Le succès de A est l'événement $\{p(x) = b^\star\}$. Cet événement se décompose selon que $p(x)$ vaut ou ne vaut pas r. Si $p(x) = r$, alors D a renvoyé 0, et si $p(x) = \overline{r}$, alors D a renvoyé 1. Donc :

$$\{p(X) = b^\star\} = \{d^\star = 0 \text{ et } p(x) = r\} \cup \{d^\star = 1 \text{ et } p(x) = \overline{r}\},$$

cette réunion étant disjointe. La probabilité de succès de A est donc

$$\Pr_{\text{succes}}(A) = \Pr(d^\star = 0 \mid p(x) = r) \times \Pr(p(x) = r) + \Pr(d^\star = 1 \mid p(x) = \overline{r}) \times \Pr(p(x) = \overline{r})$$

Le choix de r étant aléatoire, en notant que $\Pr(d^\star = 1 \mid p(x) = \overline{r}) = 1 - \Pr(d^\star = 0 \mid p(x) = \overline{r})$, et que l'avantage de A est $|2\Pr_{\text{succes}}(A) - 1|$, on obtient :

$$\mathrm{Av}(A) = \underbrace{\Pr(d^\star = 0 \mid p(x) = r)}_{\text{succès de } D} - \underbrace{\Pr(d^\star = 0 \mid p(x) = \overline{r})}_{\text{échec de } D}$$

Le premier terme ci-dessus signifie le succès du distingueur D au jeu du pseudo aléa, alors que le second terme signifie son échec. On déduit :

$$\mathrm{Av}(A) = \Big|\Pr_{\text{succes}}(D) - (1 - \Pr_{\text{succes}}(D))\Big| = \Big|2\Pr_{\text{succes}}(D) - 1\Big| = \mathrm{Av}^\star(D).$$

L'avantage de D étant supposé non négligeable, celui de A ne l'est pas non plus, ce qui achève la preuve. □

Dans le cas d'un générateur de pseudo aléa g dont l'expansion n'est que d'un seul symbole binaire, l'oracle du jeu du pseudo aléa peut aisément construire un élément aléatoire n'appartenant pas à l'image de g en tirant au hasard une valeur x, en calculant $g(x) = (y,\sigma)$ est en complémentant le second terme. La valeur transmise au distingueur est dans ce cas $(y,\overline{\sigma})$.

Exercice 2.8. *Générateur de Blum, Blum et Shub* Soient p et q deux nombres premiers qui sont congrus à 3 modulo 4. Soit $n = p \times q$. Dans l'ensemble $\mathbb{Z}/n\mathbb{Z}$, on note Q_n le sous-ensemble constitué des entiers qui sont des carrés modulo n.

1. Démontrer que la fonction carré est une bijection de Q_n. Pour tout $y \in Q_n$, on note $\sqrt{y}$ l'antécédent de y pour la fonction carré.
2. Montrer que si l'on dispose d'un oracle qui, pour tout $y \in Q_n$ donne la parité de $\sqrt{y}$, alors pour tout y, on peut calculer efficacement $\sqrt{y}$.
3. Utiliser ces propriétés pour décrire un générateur de pseudo aléa.

4.2 Générateur de pseudo aléa d'expansion polynomiale

Maintenant, à partir d'un générateur dont l'expansion est de un symbole binaire, construisons un générateur de pseudo aléa d'expansion ℓ, où ℓ reste bornée par un polynôme du paramètre de sécurité.

Théorème 4.3. Générateur de pseudo-aléa d'expansion polynomiale

Soit g la fonction définie par :

$$g : \begin{array}{ccc} \{0,1\}^n & \to & \{0,1\}^n \times \{0,1\} \\ x & \mapsto & g(x) = (y, \sigma) \end{array}$$

On suppose que cette fonction est un générateur de pseudo aléa d'expansion un symbole binaire. On pose h la fonction $\{0,1\}^n \to \{0,1\}^\ell$ définie par $h(x) = (\sigma_1, \ldots, \sigma_\ell)$ avec $x_0 = x$ et pour $i = 1$ à ℓ, la valeur de σ_i est définie par $g(x_{i-1}) = (x_i, \sigma_i)$.

Si ℓ est strictement supérieur à n et borné par un polynôme en n, alors h est un générateur de pseudo aléa.

Preuve Pour prouver ce théorème, on utilise la technique hybride déjà mise en œuvre pour la preuve du théorème 2 page 29. Pour $k = 0$ à ℓ notons H_k la variable aléatoire hybride qui consiste à choisir k symboles binaires $\sigma_1, \ldots, \sigma_k$ vraiment aléatoires, puis de générer les termes $\sigma_{k+1}, \ldots, \sigma_\ell$ à partir du germe x_k. Montrons que pour tout $k = 0$ jusqu'à $\ell - 1$, la distance calculatoire $\Delta(H_k, H_{k+1})$ est négligeable. Comme la variable H_0 est $h(U_n)$ et H_ℓ est U_ℓ, en utilisant l'inégalité triangulaire :

$$\Delta(h(U_n), U_\ell) \leq \Delta(H_0, H_1) + \cdots + \Delta(H_{\ell-1}, H_\ell),$$

cela montrera, comme chaque terme de cette somme est négligeable, et comme le nombre de termes est majoré par un polynôme en n, que la distance calculatoire $\Delta(h(U_n), U_\ell)$ reste négligeable. Par conséquent les deux variables $h(U_n)$ et U_ℓ sont calculatoirement indistinguables, montrant ainsi que la fonction h et un générateur de pseudo aléa.

Supposons pour une contradiction que la distance calculatoire $\Delta(H_k, H_{k+1})$ n'est pas négligeable, c'est-à-dire qu'il existe un distingueur D, polynomial, qui reconnaît une réalisation de H_k d'une réalisation de H_{k+1} avec un avantage non négligeable. Montrons qu'on peut utiliser ce distingueur D comme sous-programme d'un algorithme A contre le générateur de pseudo aléa g.

Algorithme 4.4. A contre le générateur de pseudo-aléa g

Entrée : un couple $(y, \sigma) \in \{0,1\}^n \times \{0,1\}$

1. choisir $\sigma_1, \ldots, \sigma_k$ aléatoires dans $\{0,1\}$
2. poser $x_{k+1} = y$ et $\sigma_{k+1} = \sigma$.
3. pour $i = k + 2$ jusqu'à ℓ, construire $(x_i, \sigma_i) = g(x_{i-1})$.
4. soumettre la séquence $\sigma_1, \ldots, \sigma_\ell$ ainsi construite au distingueur D, soit $d^\star$ sa réponse.

Sortie : renvoyer cette réponse $d^\star$.

Si l'entrée (y, σ) est pseudo aléatoire, alors l'entrée du distingueur D est une réalisation de H_k, alors que si l'entrée (y, σ) est aléatoire, il s'agit d'une réalisation de H_{k+1}. Dans ces conditions, le succès de l'algorithme A correspond exactement au succès du distingueur D. Les deux algorithmes ont exactement le même avantage. $\mathrm{Av}(D)$ étant supposé non négligeable, il en est de même pour $\mathrm{Av}(A)$, ce qui contredit l'hypothèse selon laquelle g est un générateur de pseudo aléa. □

5 Application au chiffrement

Comme affirmé dans l'introduction de ce chapitre, un générateur de pseudo aléa permet le chiffrement d'un message long avec une clé courte. Plus précisément, un générateur de pseudo aléa $g : \{0,1\}^n \to$

$\{0,1\}^\ell$ permet, avec une clé de n symboles binaires seulement, de chiffrer des messages de ℓ symboles binaires, où ℓ peut être bien plus grand que n.

— La clé secrète est le germe du générateur, soit $k \in \{0,1\}^n$.

— Pour un message de ℓ symboles binaires, le cryptogramme est $\gamma = m \oplus g(k)$.

La valeur $g(k)$ est calculatoirement indiscernable d'un véritable aléa. Elle masque donc en pratique parfaitement l'information du message m.

Ce système de chiffrement présente l'avantage, par rapport au masque jetable de Vernam, d'autoriser la taille du message à être largement supérieure à celle de la clé. Il évite la transmission préalable d'une quantité d'aléa aussi importante que les messages à chiffrer ultérieurement. L'intérêt pratique est considérable. Transmettre un aléa de 128 symboles binaires autorise le chiffrement en toute sécurité de fichiers de plusieurs mégaoctets.

Comme pour le masque jetable, on ne peut chiffrer qu'un seul message par clé. pour chiffrer un autre message, il faut une autre clé. Ce mécanisme s'apparente à ce qu'on pourrait appeler un *système de chiffrement avec clé jetable*. Les familles pseudo aléatoires de fonctions présentées au chapitre suivant permettront de proposer un mécanisme de chiffrement qui autorise le chiffrement de plusieurs messages avec une même clé.

La question de la sécurité du chiffrement sera abordée au chapitre 4.

3
Familles pseudo-aléatoire de fonctions

Nous considérons ici des fonctions d'un ensemble fini E vers un ensemble fini F. Dès que les ensembles E et F dépassent une certaine taille, il est en pratique impossible de décrire entièrement une fonction arbitraire de E vers F. Si on veut pouvoir indexer toutes les fonctions, il faut que l'information contenue dans l'index puisse entièrement décrire la table de toutes les valeurs. Cela tient à la croissance exponentielle de la taille l'ensemble $\mathcal{F}(E,F)$ de toutes les fonctions de E vers F, relativement à la taille de F. Le nombre de fonctions de l'ensemble fini E vers l'ensemble fini F vaut en effet $\mathrm{Card}(F)^{\mathrm{Card}(E)}$.

En pratique, on considère toujours des sous-familles strictes, décrites par exemple par un algorithme, ou par un paramètre d'une taille raisonnable, comme par exemple 128 symboles binaires dans le cas de la fonction AES. Mais cela ne peut décrire qu'un sous-ensemble extrêmement restreint de l'ensemble des fonctions possibles. Un index de 128 symboles binaire permet d'indexer au plus 2^{128} fonctions, alors que l'ensemble de toutes les fonctions $\{0,1\}^{32} \to \{0,1\}^{32}$ a une taille égale à $2^{32\times 2^{32}}$. Notre index de 128 symboles binaires ne peut en indexer qu'une infime partie, égale à $1/2^{32\times 2^{32}-128}$.

Une famille de fonctions est dite pseudo-aléatoire s'il est pratiquement impossible de déterminer si une fonction tirée au hasard a été tirée dans la famille ou dans l'ensemble de toutes les fonctions. Comme dans les chapitres précédents, cette difficulté est modélisée comme celle qu'a un adversaire à gagner lors d'un jeu aux règles précises. Ainsi, désigner une fonction d'une famille pseudo-aléatoire de fonctions, par exemple à l'aide d'une clé secrète, équivaut calculatoirement à choisir une fonction aléatoire.

1 Chiffrer plusieurs messages avec la même clé

Le système de chiffrement décrit à la fin du chapitre précédent permet déjà de chiffrer des messages plus longs que la clé, mais la clé ne peut servir qu'une seule fois. Une première application des familles pseudo-aléatoires de fonctions est de permettre le chiffrement de plusieurs messages avec la même clé secrète.

Soit $\mathcal{H} = (f_s)_{s\in S}$ une famille de fonctions de $\{0,1\}^n$ vers $\{0,1\}^\ell$. Une clé secrète partagée entre les correspondants est un élément s quelconque choisi aléatoirement dans l'ensemble d'indices S.

Pour chiffrer un messages m de ℓ symboles binaires, le chiffrement est calculé ainsi :

1. choisir un élément x aléatoire dans $\{0,1\}^n$;
2. le cryptogramme est constitué du couple $(x, f_s(x)\oplus m)$.

La valeur $f_s(x)$ est le masque qui dissimule l'information du message m. Ce masque peut être changé à chaque message en changeant simplement la valeur de x. La donnée x permet au destinataire, qui dispose de la connaissance du paramètre secret s, d'ôter le masque. Elle est propre à chaque message et doit être changée pour tout nouveau message. Mais la même clé peut servir à pour un grand nombre de messages, ceci au prix d'une donnée additionnelle transmise à chaque cryptogramme.

À la réception du cryptogramme $\gamma = (x,c)$ le destinataire retrouve le message en calculant $m = c \oplus f_s(x)$.

La fonction de chiffrement ainsi décrite est *non déterministe*. Comme la donnée x est choisie aléatoirement à chaque opération de chiffrement, chiffrer deux fois du même message conduit à deux cryptogrammes différents. C'est un avantage pour la sécurité, car un adversaire qui peut observer les cryptogramme ignore qu'il s'agit du même cryptogramme. Par contre, l'opération de déchiffrement est déterministe. Le message déchiffré à partir d'un cryptogramme donné est unique.

Pour assurer la confidentialité du cryptogramme, c'est-à-dire pour que le cryptogramme n'apporte aucune information sur le message en clair, le masque $f_s(x)$ doit se comporter comme un masque aléatoire, c'est-à-dire que la valeur $f_s(x)$ soit comme celle d'une fonction parfaitement aléatoire. La sécurité de ce mécanisme de chiffrement sera étudiée plus formellement au chapitre 4.

2 Indistinguabilité de familles de fonctions

On s'intéresse tout d'abord au problème général de la distinguabilité de deux familles de fonctions.

Deux familles de fonctions d'un même ensemble E vers un même ensemble F sont dites indistinguables s'il est pratiquement impossible de reconnaître un élément d'une famille d'un élément de l'autre famille. Cette propriété est formalisée par un jeu qui fait intervenir un oracle et un adversaire. L'oracle choisit un élément de l'une ou l'autre des familles, puis réponds aux requêtes de l'adversaire qui lui demande des valeurs. L'adversaire est chargé de distinguer si les valeurs qui lui sont rendues sont celles d'une fonction appartenant à l'une ou à l'autre des deux familles. Un tel adversaire et est aussi appelé un *distingueur*.

Jeu 2.1. de reconnaissance de deux familles de fonctions

Les deux joueurs disposent de la connaissance de deux familles $\mathcal{H}_0$ et $\mathcal{H}_1$ de fonctions de E vers F.

1. L'oracle choisit un symbole binaire aléatoire $b \in_R \{0,1\}$ avec la probabilité uniforme qui va indiquer dans quelle famille il va tirer une fonction.
2. L'oracle tire une fonction f aléatoire dans la famille $\mathcal{H}_b$ avec la probabilité uniforme.
3. L'adversaire soumet à l'oracle des requêtes x_i auxquelles l'oracle répond en fournissant à l'adversaire la valeur $f(x_i)$.
4. Après un certain nombre de requêtes, l'adversaire propose une estimation $b^\star$ qui indique selon lui de quelle famille la fonction f, dont il a reçu les valeurs, est issue.
5. L'adversaire a gagné si $b = b^\star$, il a perdu dans le cas contraire.

Les réponses de l'oracle aux requêtes de l'adversaire étant déterministes, deux requêtes identiques donnent des réponses identiques, soumettre une deuxième fois une même valeur n'apporte aucune information à l'adversaire. On peut donc supposer que les requêtes x_i sont deux à deux distinctes.

Le succès de l'adversaire est l'événement $\{b = b^\star\}$ et sa probabilité de succès est $\Pr(b = b^\star)$. L'avantage de l'adversaire est donné par la proposition suivante.

Proposition 2.2.

L'avantage de l'adversaire au jeu de la reconnaissance des familles de fonctions est égal à l'expression :

$$\mathrm{Av}(A) = \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 1 \mid b = 1)\Big|$$

Preuve La preuve est identique à celle de la proposition 2 page 26, étant donné que l'oracle choisit la valeur binaire b avec la probabilité uniforme. □

Exercice 3.1. Soient E et F deux espaces vectoriels sur un corps $\mathbb{F}$. Soit $\mathcal{L}$ la famille des fonctions linéaires de E vers F et $\mathcal{N}$ la famille des fonctions non linéaires. Proposer un algorithme qui gagne toujours au jeu de la reconnaissance de ces deux familles.

Pour deux familles $\mathcal{F}$ et $\mathcal{G}$ de fonctions d'un ensemble E vers un ensemble F, et un adversaire A chargé de distinguer ces deux familles au jeu de la reconnaissance, on note $\mathrm{Av}_{\mathcal{FG}}(A)$ son avantage.

Une distance calculatoire entre deux familles de fonctions se défini de façon similaire à la distance

calculatoire entre les variables aléatoires. Considérons $\mathcal{A}$ l'ensemble de tous les distingueurs des familles de fonctions $\mathcal{F}$ et $\mathcal{G}$. L'avantage d'un élément de $\mathcal{A}$ appartient à l'intervalle $[0, 1]$. L'ensemble des avantages pour tous les éléments de $\mathcal{A}$ est une partie non vide et majorée de $\mathbb{R}$, ce qui donne un sens à la définition suivante :

Définition 2.3. Distance calculatoire de deux familles de fonctions

Soient $\mathcal{F}$ et $\mathcal{H}$ deux familles de fonctions d'un ensemble E vers un ensemble F. On appelle distance calculatoire de $\mathcal{F}$ à $\mathcal{G}$ le réel égal à la borne supérieur de l'ensemble des avantages des adversaire qui les distinguent :

$$\Delta(\mathcal{F}, \mathcal{G}) = \sup_{A \in \mathcal{A}} \mathrm{Av}_{\mathcal{F}\mathcal{G}}(A)$$

Exercice 3.2. Démontrer que la distance calculatoire de deux familles de fonctions satisfait les trois axiomes d'une distance, à savoir, quelles que soient les familles de fonctions $\mathcal{F}$, $\mathcal{G}$ et $\mathcal{H}$ sur les mêmes ensembles de départ et d'arrivée :

Séparabilité : $\Delta(\mathcal{F},\ \mathcal{G}) = 0 \Leftrightarrow \mathcal{F} = \mathcal{G}$,
Symétrie : $\Delta(\mathcal{F},\ \mathcal{G}) = \Delta(\mathcal{G},\ \mathcal{F})$.
Inégalité triangulaire : $\Delta(\mathcal{F},\ \mathcal{G}) \leq \Delta(\mathcal{F},\ \mathcal{H}) + \Delta(\mathcal{H},\ \mathcal{G})$.

3 Famille de fonctions calculatoirement indistinguables

De façon informelle, deux familles de fonctions seront dites calculatoirement indistinguables, si tout adversaire polynomial n'a qu'un avantage négligeable au jeu de la reconnaissance. Cette notion est asymptotique et s'applique donc à des familles échantillonnables selon un paramètre de sécurité n, comme défini dans la définition 1, page 14.

Définition 3.1. Familles de fonctions calculatoirement indistiguables

Soient $\mathcal{F}$ et $\mathcal{G}$ deux familles de fonctions échantillonnables selon un paramètre de sécurité n dont les termes sont des fonctions $E \to F$, où la taille des éléments des E et F est majorée par un polynôme en n. On dit que ces deux familles sont calculatoirement indistinguables si tout adversaire polynomial pour le jeu de la reconnaissance a un avantage négligeable qui est une fonction négligeable de n.

Comme on se limite à des adversaires dont la complexité est polynomiale, le nombre de requêtes que cet adversaire formule à l'oracle avant de rendre sa réponse doit lui-même être borné par un polynôme en n.

Le jeu de la reconnaissance se joue pour un paramètre de sécurité n donné. On suppose que pour ce paramètre de sécurité, le domaine et les valeurs des fonctions de $\mathcal{F}$ et de $\mathcal{G}$ sont les mêmes, et donc qu'il n'est pas possible de distinguer les deux familles sur leur ensemble de départ ou d'arrivée.

Nous sommes maintenant en mesure de définir ce qu'est précisément une famille pseudo-aléatoire de fonctions. Soit $\mathcal{H}$ la famille de toutes les fonctions $\{0,1\}^\star \to \{0,1\}^\star$.

Définition 3.2. Famille pseudo-aléatoire de fonctions

On dit que la famille échantillonnable de fonctions $\mathcal{F}$ est pseudo-aléatoire si
- les valeurs des fonctions de la famille $\mathcal{F}$ sont calculables avec un algorithme efficace,
- la famille $\mathcal{F}$ est indistinguable de la famille $\mathcal{H}$ de toutes les fonctions.

On appellera souvent *fonction pseudo-aléatoire* une fonction tirée aléatoirement dans une famille pseudo-aléatoire de fonctions, alors qu'une *fonction aléatoire* est une fonction tirée aléatoirement dans l'ensemble de toutes les fonctions. Une fonction aléatoire est ainsi une fonction dont les valeurs sont aléatoires et indépendantes.

Dire que la famille $\mathcal{F}$ est une famille pseudo-aléatoire de fonction signifie qu'il n'est pas possible à un adversaire de discerner, avec un avantage non négligeable, une fonction pseudo-aléatoire, c'est-à-dire un élément aléatoire de $\mathcal{F}$, d'une fonction aléatoire définie sur le même domaine.

Exercice 3.3. La famille de fonctions $(RSA_{(m,e)})_{(m,e)\in I}$ est-elle pseudo-aléatoire ?

On peut définir la famille pseudo-aléatoire de fonctions selon un double paramétrage : d'une part le paramètre de sécurité n, et d'autre part, la valeur de n étant fixée, selon un indice s appartenant à un ensemble d'indice $\mathcal{S}$. Par exemple, pour une sécurité du RSA de 1024 symboles binaires, il est possible de produire un grand nombre de clés de cette taille. Le caractère pseudo-aléatoire de la famille se comprend asymptotiquement, lorsque le paramètre de sécurité n tend vers $+\infty$.

Exercice 3.4. *S'il existe une famille pseudo-aléatoire de fonctions alors il existe un générateur de pseudo aléa*

Pour la valeur n du paramètre de sécurité, soit $\mathcal{F}_n = (f_s)_{s\in\mathcal{S}}$ une famille pseudo-aléatoire de fonctions $\mathbb{Z}/2^n\mathbb{Z} \to \{0,1\}^n$. Montrer que pour tout élément $x \in \mathbb{Z}/2^n\mathbb{Z}$, la fonction g_x définie par :

$$g_x : \begin{array}{ccc} \mathbb{Z}/2^n\mathbb{Z} & \to & \{0,1\}^{2n} \\ s & \mapsto & \big(f_s(x), f_s(x+1)\big) \end{array},$$

est un générateur de pseudo aléa d'expansion double.

Une famille pseudo-aléatoire de fonctions est en pratique une famille d'une taille bien plus réduite que celle de la famille de toutes les fonctions. L'ensemble $\mathcal{S}$ des indices doit d'interpréter comme un ensemble de clés dans lequel deux correspondants conviennent d'un élément s qu'ils partagent en secret. Cet indice leur permet d'accéder à des valeurs $f_s(x)$ qu'eux seuls sont capable de calculer en connaissant s, y compris lorsque le paramètre x est connu de tous. Pour quiconque ignore s, la valeur $f_s(x)$ doit être est indiscernable d'une valeur aléatoire. La taille des clés peut en pratique être de 128 symboles binaires, ce qui empêche la recherche exhaustive dans l'ensemble des 2^{128} éléments. Si les correspondants devaient se mettre d'accord sur une fonction aléatoire $\{0,1\}^n \to \{0,1\}^n$, ils devraient choisir un élément parmi $2^{n\times 2^n}$. La taille des clés devrait être de $n \times 2^n$ symboles binaires, ce qui est impraticable dès que l'entier n dépasse quelques unités.

4 Famille à sens unique de fonctions

Une famille à sens unique de fonctions est une famille, indexée par un indice échantillonnable, pour laquelle il est difficile, sans connaissance de l'indice, de trouver un antécédent pour une valeur donnée.

Définition 4.1. Famille à sens unique de fonctions

On dit qu'une famille de fonctions $(f_k)_{k\in\mathcal{K}}$, où $f_k : \{0,1\}^n \longrightarrow \{0,1\}^n$, est une *famille à sens unique de fonctions* si, étant donnée un élément y de l'image $\{0,1\}^n$, et étant donné un oracle qui réalise un élément aléatoire f_k de la famille, tout algorithme efficace ne peut trouver un antécédent de y pour f_k qu'avec une probabilité négligeable.

Ne pas confondre cette notion avec celle de famille de fonctions à sens unique donnée par la définition 1 page 14. Dire qu'une famille de fonctions est à sens unique ne signifie pas que chaque élément f_k de la famille est lui-même une fonction à sens unique. Connaissant la clé k,

il n'y a aucune raison que ces fonctions soient difficiles à inverser. La définition stipule seulement que, sans connaissance de la clé, il n'est pas possible de trouver un antécédent d'un élément y alors qu'on ne dispose que d'un oracle pour déterminer les valeurs $f_k(x)$.

Exercice 3.5. Montrer que si une famille $\mathcal{F}_n$ est une famille pseudo-aléatoire de fonctions $\{0,1\}^n \to \{0,1\}^n$, alors elle est une famille à sens unique de fonctions.

On peut montrer également qu'une famille pseudo-aléatoire de fonctions $\{0,1\}^n \to \{0,1\}^n$ cache tous les composantes de l'entrée. Un adversaire polynomial ne peut déterminer une composante x_i d'un antécédent x d'une valeur y donnée qu'avec un avantage négligeable.

5 Construction de Goldreich, Goldwasser et Micali

L'exercice 4 ci-dessus montre que s'il existe une famille pseudo-aléatoire de fonctions, alors cela permet de construire un générateur de pseudo aléa d'expansion double. L'objet de cette section est de montrer la réciproque, c'est-à-dire de proposer la construction d'une famille pseudo-aléatoire de fonctions à partir d'un de générateur de pseudo aléa d'expansion double. Cela montrera l'équivalence de l'existence de ces deux notions. Cette construction a été présentée en 1984 par les chercheurs Oded Goldreich, Shafri Goldwasser et Silvio Micali.

On suppose donc que l'on dispose d'un générateur de pseudo aléa g d'expansion double, construit par exemple sur les résultats du chapitre précédent :

$$g : \begin{array}{ccc} \{0,1\}^n & \longrightarrow & \{0,1\}^{2n} \\ k & \longrightarrow & g(k) = (g_0(s), g_1(s)) \end{array}$$

On note g_0 et g_1 les fonctions dont les valeurs sont respectivement les n premières et les n dernières composantes de la valeur de la fonction g.

Pour deux indices binaires $i, j \in \{0,1\}$, notons g_{ij} la fonction $g_{ij} = g_j \circ g_i : \ x \mapsto g_j(g_i(x))$.

De même pour trois indices binaires $i, j, k \in \{0,1\}$, notons g_{ijk} la fonction $g_{ijk} = g_k \circ g_j \circ g_i$.

Généralisons pour un mot de k symboles binaires, $u = u_1 \cdots u_k \in \{0,1\}^k$, et notons g_u la fonction $g_{u_k} \circ \cdots \circ g_{u_1} : \mapsto g_{u_k}(\cdots g_{u_1}(x) \cdots)$ qui consiste à appliquer successivement $u_1, u_2, \ldots u_k$ au paramètre x.

Cette définitions définit une famille de fonctions $\mathcal{H} = (f_s)_{s \in \{0,1\}^n}$. Pour tout indice $s \in \{0,1\}^n$, posons :

$$f_s : \begin{array}{ccc} \{0,1\}^k & \to & \{0,1\}^n \\ u & \mapsto & g_u(s) \end{array} .$$

Proposition 5.1. Construction de Goldreich-Goldwasser-Micali

Si k est borné par un polynôme en n, et si g est un générateur de pseudo aléa, alors la famille $\mathcal{H}$ définie ci-dessus est une famille pseudo-aléatoires de fonctions.

Représentons cette construction par un arbre. La racine de l'arbre est le germe du générateur de pseudo-aléa, égal à l'indice s de la famille. Les deux fils de cette racine sont les valeurs $g_0(s)$ et $g_1(s)$.

Un nœud à la profondeur i dans cet arbre est constitué de l'une des 2^i valeurs $g_v(s)$, pour $v \in \{0,1\}^i$. Les fils du nœud g_v sont les deux valeurs $g_{v0}(s)$ et $g_{v1}(s)$.

Les nœuds à la profondeur k représentent les 2^k valeurs de $f_s(u) = g_u(s)$.

Une graine aléatoire à la racine de cet arbre définit un élément aléatoire de la famille $\mathcal{H}$.

Pour tout entier i compris entre 0 et k, définissons une famille de fonctions hybride $\mathcal{H}_i$ de façon similaire, mais en choisissant 2^i graines aléatoires à la profondeur k de cet arbre au lieu des valeurs $g_v(s)$, pour $v \in \{0,1\}^i$.

Plus formellement, pour un entier $i \in \{0, \ldots k\}$ et pour un mot binaire $u \in \{0,1\}^k$, décomposons $u = (v, w)$, où le préfixe $v \in \{0,1\}^i$ comprend i symboles binaires et $w \in \{0,1\}^{k-1}$ en comprend $k - i$,

définissons la famille hybride $\mathcal{H}_i$ par :

$$\mathcal{H}_i = (f_\sigma)_{\sigma=(s_v)_{v\in\{0,1\}^i}}$$

Pour un indice $\sigma = (s_v)_{v\in\{0,1\}^i}$ et un mot binaire $u = (v, w$ de $\{0,1\}^k$, la valeur $f_\sigma(u)$ est définie par :

$$f_\sigma(u) = g_w(s_v).$$

Les indices de la famille $\mathcal{H}_i$ sont constitués de 2^i $(k-i)$-uplets $s_{0\cdots0}\cdots s_{1\cdots1}$.

Le calcul des la valeur d'un élément de la famille $\mathcal{H}_i$ comprend une partie aléatoire représentée par un choix dans l'indice $\sigma = (s_v)_{v\in\{0,1\}^i}$ selon les premières composantes du paramètre x, et une partie pseudo-aléatoire qui est le résultat d'un calcul à partir l'indice choisi selon les dernières composantes de x. C'est en ce sens que cette famille est qualifiée d'hybride. La valeur $f_\sigma(u)$ est le résultat du calcul g_w sur le germe aléatoire s_v. Ainsi la famille $\mathcal{H}_0$ est la famille $\mathcal{H}$, alors que la famille $\mathcal{H}_k$ est la famille de toutes les $2^{n\times 2^k}$ fonctions $\{0,1\}^k \to \{0,1\}^n$.

Lemme suivant est le cœur de la preuve de la proposition 1 :

Lemme 5.2.

Soit i un entier compris entre 0 et $k-1$. Soit D un distingueur des familles $\mathcal{H}_i$ et $\mathcal{H}_{i+1}$ qui effectue au plus m requêtes à l'oracle. Soit G_m la variable aléatoire sur $\{0,1\}^{2n\times m}$ égale à la concaténation m fois $\underbrace{g(U_n) \mid \cdots \mid g(U_n)}_{m \text{ fois}}$, où U_n est la variable aléatoire uniforme sur $\{0,1\}^n$. Alors :

$$\mathrm{Av}(D) \leq \Delta(G_m, U_{2n\times m}),$$

où $\Delta(G_m, U_{2n\times m})$ représente la distance calculatoire entre les variables aléatoires G_m et $U_{2n\times m}$.

Preuve Utilisons le distingueur D pour construire un distingueur A des variables aléatoires G_m et $U_{2n\times m}$ avec le même avantage. Cela montrera $\mathrm{Av}(D) = \mathrm{Av}(A) \leq \Delta(G_m, U_{2n\times m})$.

Algorithme 5.3.

Entrée : Un vecteur $\alpha_1\cdots\alpha_m$ de m éléments $\alpha_\ell \in \{0,1\}^{2n}$ qui sont des soit réalisations, soit de G_m, soit des réalisation de $U_{2n\times m}$.

Soit $x \in \{0,1\}^k$ une requête de D. Décomposons $x = u \mid x_{i+1} \mid v$, où $u \in \{0,1\}^i$ est le préfixe des i premiers symboles binaires, $v \in \{0,1\}^{k-i-1}$ est le suffixe des $k-i-1$ symboles binaires, et $x_{i+1} \in \{0,1\}$ est le $(i+1)$-ième terme de x.

Si le préfixe u a déjà été joué par D à une précédente requête, alors A utilise le germe α qui a servi à répondre à cette requête.

Sinon, A utilise un nouveau germe $\alpha = \alpha_\ell$ extrait de l'entrée.

Soit $\alpha = (a_0, a_1)$ la décomposition de α en deux parties de n symboles binaires.

La valeur rendue à D est $g_v(a_{x_{i+1}})$.

Après au plus m requêtes, l'algorithme D renvoie une réponse $b^\star$ qui constitue la réponse de A.

L'algorithme A simule l'oracle auprès du distingueur D en utilisant son entrée $\alpha_1\ldots\alpha_m$. Si cette entrée est aléatoire, alors A simule exactement un oracle qui fournit des valeurs d'un élément de la famille $\mathcal{H}_{i+1}$ alors que si l'entrée est la réalisation de la variable aléatoire G_m, alors A simule exactement un oracle qui fournit des valeurs d'un élément de la famille $\mathcal{H}_i$.

Comme D utilise au plus m requêtes, l'algorithme A peut toujours répondre en utilisant des termes de l'entrée $\alpha_1\cdots\alpha_m$.

Il en résulte que le succès de D correspond exactement au succès de A. Les deux algorithmes ont par conséquent le même avantage, ce qu'il fallait prouver. □

Maintenant, prouvons la proposition 1

Preuve de la proposition 1 Après passage à la borne supérieur de l'ensemble des avantages des distingueurs des familles $\mathcal{H}_i$ et $\mathcal{H}_{i+1}$, le lemme ci-dessus permet d'établir que :

$$\Delta(\mathcal{H}_j, \mathcal{H}_{i+1}) \leq \Delta(G_m, U_{2n\times m}).$$

D'après le résultat de l'exercice 7, page 33 sur la distance calculatoire de la concaténation de m générateurs pseudo-aléatoires à la variable uniforme, on a :

$$\Delta(G_m, U_{2n\times m}) \leq m\Delta(G, U_{2n}),$$

où G désigne la variable aléatoire $g(U_n)$.

D'après l'inégalité triangulaire sur les distances calculatoires de deux familles de fonctions, on a :

$$\Delta(\mathcal{H}_0, \mathcal{H}_k) \leq \underbrace{\Delta(\mathcal{H}_0, \mathcal{H}_1) + \Delta(\mathcal{H}_1, \mathcal{H}_2) + \cdots + \Delta(\mathcal{H}_{k-1}, \mathcal{H}_k)}_{\text{tous inférieurs à } m\Delta(G, U_{2n})}$$

Finalement, on a :

$$\Delta(\mathcal{H}_0, \mathcal{H}_k) \leq k \times m \times \Delta(G, U_{2n}).$$

L'entier k ainsi que l'entier m étant majorés par un polynôme en n, il en résulte que si $\Delta(G, U_{2n})$ est négligeable, ce qui est supposé vrai par hypothèse, alors il en est de même pour $\Delta(\mathcal{H}_0, \mathcal{H}_k)$. □

Exercice 3.6. On reprend la construction de Goldreich, Goldwasser et Micali, mais au lieu de définir des fonctions sur l'ensemble $\{0,1\}^k$, avec des paramètres x qui comprennent un nombre fixé k de composantes binaires, on les définit sur $\{0,1\}^{\leq k}$. C'est-a-dire que $f(x)$ est défini pour des tailles de x variables de 1 jusqu'à k symboles binaires. Cela définit-il encore une famille pseudo-aléatoire de fonctions ?

Exercice 3.7. *Construction d'une famille pseudo-aléatoire de fonctions vectorielles à partir d'une famille de fonctions booléennes pseudo-aléatoires*

Étant donnée une famille pseudo-aléatoire $\left(f_s\right)_{s\in\mathcal{S}}$ de fonctions booléennes $f_s : \{0,1\}^n \longrightarrow \{0,1\}$, construire une famille pseudo-aléatoire de fonctions $\left(g_s\right)_{\sigma\in S'}$ de fonctions à valeurs vectorielles $g_\sigma : \{0,1\}^n \to \{0,1\}^\ell$.

4 Chiffrement

Le chiffrement est le service traditionnel offert par la cryptographie. Il a été créé pour transmettre des messages dont le contenu est illisible à quiconque l'intercepte. Aujourd'hui, la fonction première d'un système de confidentialité est d'offrir aux correspondant une façon de communiquer de manière discrète sur un canal de communication public ouvert à toutes les écoutes comme l'est le réseau internet. Cela est rendu possible par le chiffrement des messages. Celui-ci doit permettre de communiquer sans révéler la moindre information quant au message transmis. Mais pour évaluer la sécurité du procédé, c'est-à-dire sa résistance à l'attaque de briseurs de code acharnés, le système de confidentialité doit être défini de manière formelle. C'est l'objet du présent chapitre. Il permettra en particulier d'établir une preuve de la sécurité du mécanisme présenté au début du chapitre précédent.

1 Système de confidentialité

Un système de confidentialité est l'ensemble des données et des procédures qui permettent de transformer un message afin d'empêcher tout adversaire d'accéder à la moindre information qu'il porte sans la connaissance d'une clé secrète. La définition d'un tel système devra permettre d'établir qu'un adversaire polynomial n'a qu'un avantage négligeable pour retrouver une information lors d'un jeu qu'il s'agira de définir. La sécurité est donc une notion asymptotique, selon la valeur n d'un paramètre de sécurité. Ce paramètre sera le plus souvent sous-entendu et correspond en général à la taille des clés utilisées.

Définition 1.1. Système de confidentialité

Pour une valeur donnée du paramètre de sécurité, un système de confidentialité est la donnée des éléments suivants :

- Un domaine des messages en clair $\mathcal{M}$, muni d'une loi de probabilité.
- Un domaine $\mathcal{C}$ des cryptogrammes.
- Un ensemble $\mathcal{K} = \mathcal{E} \times \mathcal{D}$ de couples de clés (e, d), la clé e désigne la clé de chiffrement (*encryption*), et la clé d désigne la clé de déchiffrement (*decryption*).
  Cet ensemble de clés est échantillonnable, c'est-à-dire qu'il existe un algorithme efficace qui permet de générer un couple de clés (e, d) aléatoires.
- Une famille $(E_e)_{e\in\mathcal{E}}$ de fonctions de chiffrement efficaces $\mathcal{M} \to \mathcal{C}$,
- Une famille $(D_d)_{d\in\mathcal{D}}$ de fonctions de déchiffrement efficaces $\mathcal{C} \to \mathcal{M}$, avec la propriété fonctionnelle suivante :

$$\forall m \in \mathcal{M}\ \forall (e, d) \in \mathcal{K}\ D_d(E_d(m)) = m.$$

La propriété fonctionnelle indique que la fonction de déchiffrement, utilisée avec la clé de déchiffrement convenable, permet bien de retrouver le message en clair à partir de son cryptogramme. La question de la sécurité d'un tel système sera abordée aux paragraphes suivants.

Si la clé de chiffrement e est égale à la clé de déchiffrement d, le système est dit *symétrique*. Dans le cas contraire, il est dit *asymétrique*. Si de plus, la clé de déchiffrement ne peut pas se déduire efficacement de la clé de chiffrement, la clé de chiffrement peut sans inconvénient être publiée et le système est dit *à clé publique*.

Si la clé de déchiffrement d peut se déduire de la clé de chiffrement e par un algorithme efficace, on

peut se ramener à un système symétrique en incluant le calcul de d dans la fonction de déchiffrement.

Les messages en clair ne sont pas nécessairement équiprobables, c'est ce qu'indique la loi de probabilité sur $\mathcal{M}$. C'est par exemple le cas pour les textes en langue naturelle, définis sur l'alphabet latin. Certaines lettres peuvent être plus fréquentes que d'autres, comme le e en Français ou en Anglais, et certaines lettres peuvent s'associer ou ne pas s'associer, comme le q qui est presque toujours suivi d'un u en Français. La sécurité d'un système de confidentialité devra pouvoir être indépendant de la loi de probabilité sur $\mathcal{M}$.

La fonction de chiffrement E_e peut ne pas être déterministe. Cela signifie que le chiffrement deux fois d'un même message peut conduire à des cryptogrammes différents. C'est par exemple le cas pour le chiffrement ElGamal. Par contre la fonction de déchiffrement D_d doit renvoyer l'unique message en clair possible. Elle est nécessairement déterministe.

Il existe deux façons d'évaluer la sécurité d'un système de confidentialité : la *sécurité sémantique* et l'*indistinguabilité*. La première est plus intuitive, alors que la seconde est plus opératoire pour établir les preuves. Ces deux notions sont définies dans les paragraphes qui suivent. Le résultat principal de ce chapitre est de montrer l'équivalence de ces deux notions.

2 La sécurité sémantique

En cryptographie, la sécurité parfaite signifie que le cryptogramme n'apporte aucune information sur le message en clair. La sécurité sémantique en est la version calculatoire. Dans un système de confidentialité sémantiquement sûr, tout ce qui est calculable sur le clair peut se calculer sans le cryptogramme. Le cryptogramme ne permet pas de calculer la moindre information sur le message en clair qu'on ne saurait calculer sans celui-ci. *A contrario*, si le cryptogramme permet d'extraire une certaine valeur binaire du clair, alors la sécurité sémantique n'est pas assurée.

La sécurité sémantique s'évalue à l'aune de l'avantage qu'a un adversaire à gagner à un jeu appelé *jeu de la sécurité sémantique*. Ce jeu oppose un oracle à un adversaire.

Jeu 2.1. de la sécurité sémantique

Soit $b : \mathcal{M} \to \{0,1\}$ une fonction d'information sur l'ensemble des messages.

1. L'oracle choisit un message $m \in \mathcal{M}$ selon la loi de probabilités sur $\mathcal{M}$. Ce message est tenu secret.
2. L'oracle choisit un couple de clés (e,d) aléatoire dans $\mathcal{K}$.
3. L'oracle présente à l'adversaire le cryptogramme $\gamma = E_d(m)$.
4. L'adversaire renvoie à l'oracle une estimation $b^\star$ de la valeur de $b(m)$.

L'adversaire gagne si $b^\star = b(m)$. Il perd dans le cas contraire.

Pour gagner à ce jeu, on n'exige pas que l'adversaire puisse déchiffrer entièrement le défi, mais seulement d'énoncer une information binaire sur le message en clair, comme par exemple dire à partir du cryptogramme si le message en clair contient ou non le mot « *attaque* ».

Pour que ce jeu ait un sens, la longueur du cryptogramme ne doit pas révéler d'information sur la valeur de l'information $b(m)$. Pour simplifier, on se limite à des cryptogrammes de taille fixe ou imposée.

De même, la valeur de la fonction d'information ne doit pas être prévisible, ou du moins elle ne doit l'être qu'avec un avantage négligeable. Cela signifie que l'adversaire n'a pas d'intérêt à choisir 0 plutôt que 1 comme valeur. Cela conduit à définir ce qu'est une fonction d'information quasi équilibrée.

Définition 2.2. Fonction d'information quasi équilibrée

Une fonction d'information $b : \mathcal{M} \to \{0,1\}$ est quasi équilibrée si la fonction :

$$n \mapsto \Big|\Pr(b(m)=0) - \Pr(b(m)=1)\Big|,$$

où n est le paramètre de sécurité, est négligeable.

Cette notion dépend étroitement de la loi considérée sur l'ensemble des messages $\mathcal{M}$. Une fonction d'information peut être quasi équilibrée pour une certaine loi de probabilité, et ne pas l'être pour une autre loi.

Exercice 4.1. Soient m_0 et m_1 deux messages aléatoires indépendants de $\mathcal{M}$. Montrer que si la fonction $b : \mathcal{M} \to \{0,1\}$ est quasi équilibrée, alors la quantité $\Big|1 - 2\Pr\big(b(m_0) = b(m_1)\big)\Big|$ est négligeable.

Le jeu de la sécurité sémantique permet de définir ce qu'est un système de confidentialité sémantiquement sûr.

Définition 2.3. Système de confidentialité sémantiquement sûr

On dit qu'un système de confidentialité est sémantiquement sûr si pour toute loi de probabilité sur $\mathcal{M}$ et toute fonction d'information $b : \mathcal{M} \to \{0,1\}$ quasi équilibrée relativement à cette loi, l'avantage de tout adversaire polynomial au jeu de la sécurité sémantique est négligeable.

La sécurité sémantique signifie que l'adversaire ne peut évaluer la moindre information sur le message en clair à partir du cryptogramme qu'avec un avantage négligeable.

Exercice 4.2. Le chiffrement à clé publique $\mathrm{RSA}_{e,n}$ défini par $\gamma = m^e \bmod n$, est-il sémantiquement sûr ?

3 Indistinguabilité

La sécurité sémantique d'un système de confidentialité est délicate à établir, car elle doit l'être pour toute loi de probabilité sur l'ensemble des messages et toute fonction d'information quasi équilibrée. Pour cette raison, une autre notion, équivalente et plus facile à manipuler est introduite, celle d'indistinguabilité.

Cette notion repose aussi sur un jeu qui oppose un oracle à un adversaire. Les deux joueurs connaissent le système de confidentialité.

Jeu 3.1. de l'indistinguabilité

1. L'oracle choisit un couple de clés (e,d) dans $\mathcal{K}$.
2. L'adversaire choisit deux messages m_0 et m_1 dans $\mathcal{M}$ qu'il transmet à l'oracle.
3. L'oracle choisit une valeur binaire aléatoire $b \in_{\mathrm{R}} \{0,1\}$, chiffre le message m_b et renvoie à l'adversaire le cryptogramme correspondant.
4. L'adversaire doit estimer de quel message m_0 ou m_1 le cryptogramme est issu, et pour cela propose une valeur $b^\star$ de b.

L'adversaire a gagné si $b^\star = b$, il a perdu dans le cas contraire.

Rappelons que l'avantage de l'adversaire A à ce jeu est donné par :

$$\mathrm{Av}_{IND}(A) = \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1)\Big|$$

Pour que ce jeu présente un intérêt, il faut que les messages m_0 et m_1 soient de même taille, ou du moins que la taille du cryptogramme de donne pas d'information sur le message qui a été chiffré par l'oracle. On supposera pour simplifier que tous les messages et tous les cryptogrammes ont la même taille.

Lorsque l'adversaire ne peut savoir quel message a été chiffré qu'avec un avantage négligeable, le système de chiffrement est dit indistinguable, ou bien qu'il a la propriété d'indistinguabilité.

Définition 3.2. Système de confidentialité indistinguable

On dit qu'un système de confidentialité a la propriété d'indistinguabilité si l'avantage de tout adversaire polynomial au jeu de l'indistinguabilité est négligeable.

Si un chiffrement à clé publique est déterministe, c'est-à-dire si la clé de chiffrement est connue de tous, en particulier de l'adversaire, et le chiffrement d'un message conduit toujours au même cryptogramme, alors un adversaire peut toujours chiffrer les deux messages qu'il a choisi et observer quel est des deux cryptogrammes celui qui lui a été transmis par l'oracle. Cette stratégie est toujours gagnante. Par conséquent :

Proposition 3.3.

Un chiffrement à clé publique déterministe n'a jamais la propriété d'indistinguabilité.

En d'autres termes et de façon équivalente, pour qu'un système de confidentialité à clé publique soit indistinguable, il est nécessaire qu'il soit non déterministe.

Exercice 4.3. Soit un système de confidentialité Σ défini par une fonction de chiffrement $E_e(m) : \{0,1\}^n \to \{0,1\}^n$. On change légèrement la fonction de chiffrement en posant :

$$\widetilde{E}_e(m) = \begin{cases} E_e(m) & \text{si} \quad m \neq 0^n, 1^n \\ 0^n & \text{si} \quad m = 0^n \\ 1^n & \text{si} \quad m = 1^n \end{cases}$$

a) Le nouveau système de confidentialité $\widetilde{\Sigma}$ est-il indistinguable ?
b) Est-il sémantiquement sûr ?

Exercice 4.4. *Indistinguabilité du mode ECB*

On dispose d'un système de chiffrement défini par une fonction de chiffrement E_e opérant sur des blocs de n symboles binaires. Le mode ECB (*Electronic CodeBook*) consiste à chiffrer des messages constitués de plusieurs blocs en chiffrant individuellement et séparément chaque bloc :

$$\widetilde{E}_e(m_1, \ldots, m_k) = E_e(m_1) \mid \cdots \mid E_e(m_k)$$

Ce mode ECB d'un chiffrement par blocs a-t-il la propriété d'indistinguabilité ?

4 Équivalence de la sécurité sémantique et de l'indistinguabilité

Le principal résultat de ce chapitre est le théorème suivant qui énonce l'équivalence entre la sécurité sémantique et la propriété d'indistinguabilité.

Théorème 4.1. Équivalence entre la sécurité sémantique et l'indistinguabilité

Un système de confidentialité est sémantiquement sûr si et seulement si il a la propriété d'indistinguabilité.

Preuve Pour un système de confidentialité Σ quelconque, notons IND la proposition « *le système* Σ *est indistinguable* » et notons SEM la proposition « *le système* Σ *est sémantiquement sûr.* » Montrons tout d'abord l'implication $SEM \Rightarrow IND$. Pour cela, montrons la contraposée de cette implication : $\neg IND \Rightarrow \neg SEM$.

Par hypothèse, il existe deux messages, m_0 et m_1 dont un adversaire polynomial A sait distinguer le cryptogramme avec un avantage non négligeable. Montrons alors que le système de confidentialité n'est pas sémantiquement sûr, c'est-à-dire qu'il existe une distribution de probabilité et une fonction d'information sur l'ensemble des messages $\mathcal{M}$ qu'un adversaire polynomial sait retrouver avec un avantage non négligeable. Considérons la loi de probabilité suivante sur l'ensemble $\mathcal{M}$ des messages :

$$\Pr(m) = \begin{cases} \frac{1}{2} & \text{si } m = m_0 \text{ ou } m_1 \\ 0 & \text{sinon} \end{cases}$$

et la fonction d'information :

$$b(m_0) = 0,\ b(m_1) = 1 \quad \text{et } b(m) \quad \text{quelconque si } m \neq m_0, m_1.$$

Cette fonction d'information est clairement quasi équilibrée pour la loi de probabilité définie sur $\mathcal{M}$.

L'adversaire A au jeu de l'indistinguabilité peut aussi jouer au jeu de la sécurité sémantique avec la loi de probabilité et la fonction d'information définie ci-dessus. Dans ce cas, il reçoit, tout comme au jeu de l'indistinguabilité, le cryptogramme d'un message qui est, soit celui de m_0, soit celui de m_1 avec probabilité égale. Soit m_ε, avec $\varepsilon \in \{0,1\}$ le message qui a été chiffré. La réponse de A est une estimation de la valeur de ε, soit $b^\star = 0$ s'il estime avoir a reçu le cryptogramme de m_0 et $b^\star = 1$ s'il estime avoir reçu le cryptogramme de m_1. Mais par définition de la fonction d'information b, cette valeur de $b^\star$ est aussi l'estimation de $b(m_\varepsilon)$.

Le succès de l'algorithme A au jeu de la sécurité sémantique est exactement son succès au jeu de l'indistinguabilité. Il en résulte que l'adversaire A a le même avantage aux deux jeux. Comme son avantage est non négligeable par hypothèse, le système de confidentialité n'est pas sémantiquement sûr.

Montrons maintenant la réciproque, c'est-à-dire si un système de confidentialité est indistinguable, alors il est sémantiquement sûr. Pour cela, montrons la contraposée de cette implication : $\neg SEM \Rightarrow \neg IND$.

Par hypothèse, il existe une loi de probabilité et une fonction d'information b quasi équilibrée sur l'ensemble des messages, ainsi qu'un adversaire A qui a un avantage non négligeable au jeu de la sécurité sémantique. Utilisons cet adversaire pour construire un adversaire B qui a un avantage non négligeable au jeu de l'indistinguabilité

Algorithme 4.2. Adversaire B au jeu de l'indistinguabilité

- B choisit deux messages aléatoires m_0 et m_1 selon la loi de probabilité sur $\mathcal{M}$ et fournit ces deux messages à l'oracle.
- B reçoit de l'oracle le cryptogramme $c_{\widetilde{b}}$ de $m_{\widetilde{b}}$, pour une valeur de $\widetilde{b} \in \{0,1\}$ choisie aléatoirement par l'oracle et inconnue de B.
- B transmet ce cryptogramme à l'adversaire A qui lui renvoie une estimation $\widehat{b}$ de la fonction d'information sur m_b.
- Si $\widehat{b} = b(m_0)$ alors renvoyer $b^\star = 0$,
- Si $\widehat{b} = b(m_1)$ alors renvoyer $b^\star = 1$.
- Et sinon renvoyer une valeur $b^\star$ aléatoire de $\{0,1\}$.

L'heuristique qui guide cet algorithme est la suivante : si A a trouvé la bonne valeur de la fonction d'information sur le message en clair, alors cela permet de discriminer entre les messages m_0 et m_1. Sachant par hypothèse que l'avantage de A est non négligeable, montrons que l'avantage de B est aussi non négligeable.

L'expression de l'avantage de B est :

$$\mathrm{Av}(B) = \Big|\Pr(b^\star = 0 \mid \widetilde{b} = 0) - \Pr(b^\star = 0 \mid \widetilde{b} = 1)\Big|.$$

L'événement $\{b^\star = 0\}$ survient dans deux cas : tout d'abord si $b(m_0) = \widehat{b}$, ou encore avec probabilité $1/2$ si les valeurs $b(m_0)$ et $b(m_1)$ sont toutes deux différentes de $\widehat{b}$. Par conséquent :

$$\Pr(b^\star = 0 \mid \widetilde{b} = 0) = \Pr(\widehat{b} = b(m_0) \mid \widetilde{b} = 0) + \frac{1}{2}\Pr(\widehat{b} \neq b(m_0), b(m_1) \mid \widetilde{b} = 0) \tag{4.1}$$

De même,

$$\Pr(b^\star = 0 \mid \widetilde{b} = 1) = \Pr(\widehat{b} = b(m_0) \mid \widetilde{b} = 1) + \frac{1}{2}\Pr(\widehat{b} \neq b(m_0), b(m_1) \mid \widetilde{b} = 1) \tag{4.2}$$

Dans l'événement $\{\widehat{b} \neq b(m_0), b(m_1)\}$, les deux messages m_0 et m_1 jouent des rôles symétriques, la probabilité de cet événement est donc indépendante de la valeur de $\widetilde{b}$:

$$\Pr(\widehat{b} \neq b(m_0), b(m_1) \mid \widetilde{b} = 0) = \Pr(\widehat{b} \neq b(m_0), b(m_1) \mid \widetilde{b} = 1)$$

Par différence des équations 4.1 et 4.2, on a :

$$\mathrm{Av}(B) = \Big|\Pr(\widehat{b} = b(m_0) \mid \widetilde{b} = 0) - \Pr(\widehat{b} = b(m_0) \mid \widetilde{b} = 1)\Big|$$

Le premier terme est la probabilité de succès de l'adversaire A. Comme m_0 est aléatoire et indépendant de m_1, le second terme est la probabilité que A réponde la valeur de la fonction d'information d'un message aléatoire indépendant de son entrée. Au coefficient de normalisation $1/\Pr(\widehat{b} \neq b(m))$ près, il s'agit, par la définition 2 page 16, de l'avantage de A. Donc :

$$\mathrm{Av}(B) = \mathrm{Av}(A) \times \Pr(\widehat{b} \neq b(m)),$$

où m est un message aléatoire selon la loi de probabilité sur $\mathcal{M}$.

La fonction d'information b étant quasi équilibrée, pour toute valeur η strictement inférieure à $1/2$, par exemple $\eta = 1/3$, et pour une valeur du paramètre de sécurité n assez grande, on a $\Pr(\widehat{b} = b(m)) \geq \eta$, donc $\mathrm{Av}(B) \geq \eta\,\mathrm{Av}(A)$. Cette inégalité montre que l'avantage de l'adversaire B est non négligeable. □

5 Modèles d'attaque

Avant de donner sa réponse, l'adversaire peut disposer d'informations pour l'aider à résoudre son problème. Les attaques sont classées selon le type d'information auquel l'adversaire a accès.

Attaque à clairs choisis, CPA *Chosen Plaintext Attack* L'adversaire peut demander à un oracle le chiffrement de messages de son choix. Dans le cas d'un chiffrement à clé publique, la clé de chiffrement étant connue, cette opération est toujours possible et ne nécessite pas d'oracle particulier. La résistance à une attaque à clairs choisis est le minimum qu'on puisse exiger d'un système de confidentialité à clé publique.

Attaque à cryptogrammes choisis, CCA *Chosen Ciphertext Attack* L'adversaire peut demander le déchiffrement de messages de son choix, sauf bien sûr le déchiffrement du défi à résoudre, tant lors du jeu de la sécurité sémantique que lors du jeu de l'indistinguabilité.

Attaque à clairs connus, KPA *Known Plaintext Attack* L'adversaire dispose de la connaissance d'un certain nombre de couples clair-cryptogramme non choisis par lui, par exemple certains déchiffrement de messages antérieurement émis.

Attaques adaptatives - non adaptatives L'attaque est dite *non adaptative* lorsque l'adversaire n'a accès à l'oracle – de chiffrement ou de déchiffrement – qu'avant la communication du défi.

L'attaque est dite *adaptative* lorsque l'adversaire a accès à l'oracle pendant tout le déroulement du jeu, y compris après la communication du défi. Il peut alors formuler ses requêtes en fonction du défi qu'il a reçu.

Taxinomie des attaques Cette classification des attaques relativement au jeu qui définit la sécurité – indistinguabilité ou sécurité sémantique – ou relativement au modèle d'attaque – clairs connus, clair choisis ou cryptogrammes choisis – s'exprime par des abréviations, par exemple :

- Un système de confidentialité est dit IND-CPA s'il a la propriété d'indistinguabilité lors d'une attaque à clairs choisis.
- Un système de confidentialité est dit SEM-CCA s'il est sémantiquement sûr contre une attaque à cryptogrammes choisis.

Lors d'une attaque à clairs choisis, l'adversaire peut requérir le chiffrement de messages de son choix. En particulier, au cours du jeu de l'indistinguabilité, il n'est pas interdit qu'il puisse demander à chiffrer les messages m_0 et m_1 qu'il a transmis à l'oracle en début de partie. En conséquence, si le chiffrement est déterministe il est aisé de construire un adversaire qui gagne à tout coup. Dans ces conditions, un chiffrement déterministe n'est jamais IND-CPA. Pour de tels chiffrements, on peut toutefois redonner un intérêt au jeu en interdisant à l'adversaire de demander le chiffrement des messages m_0 et m_1 qu'il a lui-même présenté.

Exercice 4.5. *Un chiffrement IND-CPA de messages de taille fixe à partir d'une famille pseudo-aléatoires de fontions*

Soit $\mathcal{H} = (f_s)_{s\in\mathcal{S}}$ une famille pseudo-aléatoire de fonctions $\{0,1\}^n \to \{0,1\}^k$. On considère le système de confidentialité suivant :

- L'ensemble des messages est $\mathcal{M} = \{0,1\}^k$.
- L'ensemble des cryptogrammes est l'ensemble des couples $\mathcal{C} = \{0,1\}^n \times \{0,1\}^k$.
- Le système est symétrique et l'ensemble des clés est l'ensemble $\mathcal{S}$ des indices de la famille de fonctions.
- La fonction de chiffrement est $E_s : \ m \mapsto \big(r, m \oplus f_s(r)\big)$, où r est une donnée aléatoire de $\{0,1\}^n$.

a) Décrire une fonction de déchiffrement.
b) Démontrer que ce système de confidentialité n'est pas IND-CCA.
c) Démontrer que ce système de confidentialité est IND-CPA.

Exercice 4.6. *Un chiffrement IND-CPA de messages de taille variable*

Soit $(f_s)_{s\in\mathcal{S}}$ une famille pseudo-aléatoire de fonctions $\mathbb{Z}/2^n\mathbb{Z} \to \{0,1\}^k$. On considère le système de confidentialité défini par :

- L'ensemble des messages est un ensemble $\mathcal{M}$, inclus dans $\{0,1\}^{k\star}$, constitué de blocs de k symboles binaires en nombre majoré par un polynôme en n.
- L'ensemble des cryptogrammes est l'ensemble des couples $\mathcal{C} = \mathbb{Z}/2^n\mathbb{Z} \times \{0,1\}^{k\star}$.
- Le chiffrement est symétrique et son ensemble des clés est l'ensemble $\mathcal{S}$ des indices de la famille de fonctions.
- La fonction de chiffrement est $E_s : \ m = m_1 \cdots m_\ell \mapsto \big(r, m_1 \oplus f_s(r+1) \mid \cdots \mid m_\ell \oplus f_s(r+\ell)\big)$. Chaque terme $f_s(r+j)$ constituant un masque pour le terme m_j du message en clair.

a) Décrire une fonction de déchiffrement.
b) Ce système de chiffrement est-il IND-CCA ?
c) Démontrer que ce système de chiffrement est IND-CPA.
Indication : Construire un adversaire contre la famille $(f_s)_{s\in\mathcal{S}}$ qui exploite un adversaire A au jeu de l'indistinguabilité qui peut solliciter un oracle de chiffrement. Pour évaluer son avantage considérer l'événement E « *un des termes d'un cryptogramme sollicité par A a été masqué par une même valeur que l'un des terme du défi* ».

Ce type de chiffrement est un mode opératoire standardisé des chiffrements par blocs sous le nom de *mode compteur*, ou *mode CTR*. Il est décrit dans la publication SP800-38A du NIST (*National Institute of Standard and Technology*) du département américain du commerce. Il est prouvé sûr contre une attaque à clairs choisis, y compris lorsque la fonction de chiffrement f_s n'est pas inversible. Il est assimilable à un *stream cipher*, dans lequel le flux binaire du clair est masqué par le flux binaire pseudo-aléatoire constitué des valeurs successives des masques.

5 Codes d'authentification

Les systèmes de confidentialité présentés jusqu'à présent ne résistent pas à une attaque à cryptogrammes choisis. La raison est qu'il est possible de manipuler les cryptogrammes, de les soumettre à l'oracle de déchiffrement et ainsi d'obtenir des relations sur les clairs correspondants et en tirer un bénéfice pour répondre au défi posé.

Par exemple, si la fonction de chiffrement est donnée par :

$$E_s(m) = (r, m \oplus f_s(r)),$$

où f_s est une fonction pseudo-aléatoire et où la première composante r du cryptogramme est choisi au hasard. L'élément r constitue une clé de message qui permet au destinataire de lever le masque $f_s(r)$ qui pèse sur le message. Deux cryptogrammes distincts construits avec la même clé de message peuvent révéler des informations importantes sur leurs clairs respectifs. En effet les messages m et m' dont les cryptogrammes respectifs sont (r, c) et (r, c') ont entre eux la relation suivante :

$$c \oplus m = c' \oplus m' = f_s(r)$$

Ainsi, un adversaire pourra requérir le déchiffrement d'un cryptogramme modifié pour extorquer des informations sur le message en clair du défi et donner ainsi une réponse certaine au jeu, tant de l'indistinguabilité que de la sécurité sémantique.

Cette propriété de pouvoir manipuler les cryptogrammes pour en faire d'autres cryptogrammes valides s'appelle la *malléabilité*. Pour qu'un chiffre résiste aux attaque à cryptogrammes choisis, il est nécessaire que le cryptogramme ne soit pas malléable. Changer la moindre parcelle d'un cryptogramme ne doit pas permettre d'invoquer l'oracle de déchiffrement et espérer ainsi récupérer une information sur le message en clair du défi. Une façon de procéder est d'adjoindre au cryptogramme une empreinte qui assure son authenticité et rend ainsi toute modification non authentique. Pour satisfaire cette exigence, l'empreinte aussi doit dépendre d'une clé secrète. Si l'authenticité n'est pas assurée, alors le cryptogramme pourra être rejeté, il ne sera pas déchiffré, brisant tout espoir d'accéder à la moindre information. La primitive qui assure cette fonction s'appelle un *code d'authentification* ou MAC (*Message Authentication Code*). L'objet de ce chapitre est l'étude de la sécurité, la construction et l'application de tels codes à la construction d'un système de confidentialité résistant aux attaques à cryptogrammes choisis.

1 Définition

Un code d'authentification est une primitive cryptographique qui calcule une empreinte d'un message à l'aide d'une clé secrète. L'empreinte dépend étroitement à la fois du message et de la clé de sorte qu'elle ne peut être produite que par leurs détenteurs et que toute modification du message la rend invalide. Il s'agit en quelque sorte d'une signature à clé symétrique où la clé de signature est la même que la clé de vérification. Ce type de mécanisme assure à la fois l'intégrité du message et l'authentification de son émetteur dans la mesure où la production d'une empreinte valide n'est possible qu'en connaissant un secret. Mais comme la clé est partagée par le producteur de l'empreinte et par le vérificateur, les empreintes produites par un tel code sont répudiables. Il ne sera pas possible de prouver que c'est bien le signataire annoncé qui a produit l'empreinte du document et non pas un vérificateur malhonnête. De plus, contrairement à une véritable signature, la clé de vérification n'est pas publique. Seuls les détenteurs de la clé peuvent vérifier l'empreinte.

Définition 1.1. Code d'authentification de message

Pour une valeur donnée n du paramètre de sécurité, un code d'authentification de message (MAC) est la donnée de :

- Un ensemble $\mathcal{M}$ des messages.
- Un ensemble $\mathcal{C}$ des codes.
- Un domaine des clés $\mathcal{K}$.
  Cet ensemble des clés est échantillonnable, c'est-à-dire il existe un algorithme efficace qui permet de générer une clé aléatoire dans l'ensemble $\mathcal{K}$ des clés.
- Une famille $(C_k)_{k\in\mathcal{K}}$ de fonctions efficaces de productions du code $\mathcal{M} \to \mathcal{K}$.
- Une famille $(V_k)_{k\in\mathcal{K}}$ de fonctions efficaces de vérification du code :

$$V_k : \begin{array}{ccc} \mathcal{M}\times\mathcal{C} & \to & \{0,1\} \\ (m,c) & \mapsto & v \end{array}$$

telle que :

$$\forall m \in \mathcal{M}\ \forall k \in \mathcal{K}\ V_k(m, C_k(m)) = 1$$

La valeur 1 de la vérification signifie son succès, et une valeur 0 signifie son échec. La condition sur la fonction de vérification signifie que la vérification d'une empreinte correctement générée est toujours un succès.

Comme la clé est la même pour produire et pour vérifier l'empreinte, et si la fonction de génération d'empreinte est déterministe, on peut toujours supposer que la vérification se fait en recalculant l'empreinte et en comparant le résultat avec celle qui est associée au message.

Définition 1.2. procédure de vérification générique

On appelle procédure de vérification générique du message et de son empreinte (m, c) la procédure suivante :

1. calculer $c^\star = C_k(m)$.
2. si $c = c^\star$ renvoyer 1, sinon renvoyer 0.

Cette procédure de vérification générique ne fonctionne bien sûr pas si la fonction de génération de code n'est pas déterministe, c'est-à-dire si le calcul du code donne des codes distincts lorsqu'il est appelé deux fois pour le même message.

2 Sécurité

La sécurité d'un code d'authentification de message se mesure à la difficulté de forger une empreinte valide lors d'une attaque à messages choisis. Cette sécurité est formalisée par un jeu au cours duquel l'adversaire a pour objectif de construire un message assorti d'une empreinte qui sera validée par l'algorithme de vérification, et ceci sans disposer de la clé secrète qui permet de produire ces empreintes. Pour répondre à cet objectif, l'adversaire peut interroger un oracle qui lui fournira des empreintes valides de certains messages de son choix. On parle alors d'attaque à messages choisis.

Bien sûr pour que le jeu présente un intérêt, le message que l'adversaire fournira à la fin du jeu doit être différent de tous les messages qu'il a auparavant adressés à l'oracle.

Jeu 2.1. de l'authentification

1. L'oracle choisit une clé secrète k, aléatoire dans $\mathcal{K}$
2. L'adversaire soumet à l'oracle des messages m_i, ce dernier lui répond avec $c_i = C_k(m_i)$.
3. Au bout d'un temps polynomial, l'adversaire fournit un couple message-code $(m, c) \in \mathcal{M} \times \mathcal{C}$, où m est différent de tous les m_i qu'il a soumis du cours de la phase 2.

L'adversaire a gagné si $V_k(m, c) = 1$. Il a perdu dans le cas contraire.

La performance d'un adversaire se mesure à sa probabilité de succès au jeu de l'authentification :

$$\Pr_{\text{succes}}(A) = \Pr\left(V_k(m, c) = 1\right).$$

La qualité d'un code d'authentification χ se mesure à meilleur performance d'un adversaire. Elle est la borne supérieure de l'ensemble des probabilités de succès de tous ses adversaires. En notant $\mathcal{A}$ l'ensemble de tous les adversaires au jeu de l'authentification, l'insécurité du MAC est donnée par :

$$\text{Insec}(\chi) = \sup_{A \in \mathcal{A}} \Pr_{\text{succes}}(A).$$

L'insécurité d'un code d'authentification est un nombre réel compris entre 0 et 1. Une insécurité nulle correspond à un code de très haute sécurité. Elle est d'autant plus élevée qu'il existe des adversaires qui ont une forte probabilité de gagner au jeu de l'authentification, et donc que le code présente une faible sécurité.

Par ailleurs, un adversaire peut toujours choisir une empreinte au hasard. Une autre stratégie est de choisir une clé au hasard et d'espérer qu'elle sera celle choisie par l'oracle. Ainsi, si les ensembles $\mathcal{C}$ des codes et $\mathcal{K}$ des clés sont finis, et si chaque code ou chaque clé a la même probabilité d'apparition, l'insécurité d'un code d'authentification χ satisfait les minorations suivantes :

$$\text{Insec}(\chi) \geq \frac{1}{\text{Card}(\mathcal{C})} \quad \text{et} \quad \text{Insec}(\chi) \geq \frac{1}{\text{Card}(\mathcal{K})}$$

Définition 2.2. MAC sûr

On dit qu'un code d'authentification est sûr si son insécurité est une fonction négligeable du paramètre de sécurité.

Une famille pseudo-aléatoire de fonctions définit un code d'authentification de messages sûr, comme le montre l'exercice suivant :

Exercice 5.1. Soit $\mathcal{F} = (f_s)_{s \in S}$ une famille échantillonnable de fonctions $E \to \{0,1\}^n$, où n est la valeur du paramètre de sécurité. Soit χ le code d'authentification défini ainsi :

- L'ensemble des messages est l'ensemble de départ E des éléments de $\mathcal{F}$.
- L'ensemble des codes est l'ensemble d'arrivée $\{0,1\}^n$ des éléments de $\mathcal{F}$.
- Le domaine des clés est l'ensemble $\mathcal{S}$ des indices.
- La fonction de génération d'empreinte avec la clé s est la fonction f_s.
- La fonction de vérification est la fonction de vérification générique.

Démontrer que si la famille $\mathcal{F}$ est une famille pseudo-aléatoire de fonctions, alors le code d'authentification χ est sûr.

3 Le CBC-MAC

Le code d'authentification présenté dans l'exercice ci-dessus est d'un intérêt limité. Il ne fonctionne que sur un ensemble de messages réduits à l'ensemble E de départ des éléments de la famille $\mathcal{F}$. Le

CBC-MAC permet de produire des empreintes de taille fixe pour des messages bien plus longs, constitués de plusieurs blocs de symboles binaires.

Définition 3.1. CBC-MAC

Soit $\mathcal{F} = (f_s)_{s\in\mathcal{S}}$ une famille pseudo-aléatoire de fonctions $\{0,1\}^n \to \{0,1\}^n$. On appelle CBC-MAC le code d'authentification défini par les éléments suivants :

- L'ensemble des messages est $\{0,1\}^{\ell\times n}$.
- L'ensemble des codes est $\{0,1\}^n$.
- Le domaine des clés est l'ensemble $\mathcal{S}$ des indices de la famille $\mathcal{F}$.
- L'empreinte du message $(m_1,\ldots,m_\ell$ est la valeur c_ℓ, où $c_0 = 0$ et pour tout i compris entre 1 et ℓ, on a $c_i = f_s(m_i \oplus c_{i-1})$.
- La fonction de vérification est la fonction de vérification générique.

Le code c_i issu du bloc de message m_i est chaîné au bloc de message suivant pour calculer le code suivant, de façon similaire au mode opératoire CBC (*Cipher Block Chaining*) des chiffrements par bloc.

Contrairement au mode d'opération CBC des chiffrements par blocs, le CBC-MAC ne nécessite pas que les éléments f_s de la famille soient des permutations. De simples fonctions pseudo-aléatoires conviennent.

Le théorème suivant énonce la sécurité du CBC-MAC.

Théorème 3.2. Sécurité du CBC-MAC

Si la famille $\mathcal{F}$ est pseudo-aléatoire alors le CBC-MAC, qui opère sur des messages de taille fixe, constitués de ℓ blocs de n symboles binaires, est sûr.

Preuve Supposons que le CBC-MAC n'est pas sûr. Cette hypothèse implique l'existence d'un adversaire A qui sait forger l'empreinte d'un message. Utilisons cet adversaire pour construire un adversaire polynomial B contre la famille $\mathcal{F}$ qui a un avantage non négligeable, ce qui établira le résultat du théorème. Cet adversaire B qu'il nous faut construire joue contre un oracle qui a tiré aléatoirement un symbole binaire b dans l'ensemble $\{0,1\}$. Si $b = 0$ cet oracle tire une fonction $f = f_s$ dans la famille $\mathcal{F}$ et si $b = 1$, il tire une fonction f parmi toutes les fonctions $\{0,1\}^n \to \{0,1\}^n$. L'oracle répond aux requêtes $x \in \{0,1\}^n$ de l'algorithme B en renvoyant la valeur $y = f(x)$. L'objectif de B est de renvoyer une estimation $b^\star$ de b. L'adversaire B va utiliser l'adversaire A du jeu de l'authentification, et pour cela, il va simuler un oracle d'authentification.

Algorithme 3.3. B contre la famille $\mathcal{F}$

1. Lorsque A soumet des messages m^i à signer, B s'adresse à l'oracle pour obtenir les valeurs nécessaires au calcul de l'empreinte.
2. Après un nombre q de soumissions, l'algorithme A propose un message m et son empreinte $\tau^\star$.
3. L'algorithme B s'adresse à l'oracle pour obtenir les valeurs nécessaires au calcul de l'empreinte τ du message m.
4. Si $\tau = \tau^\star$, alors B renvoie la valeur $b^\star = 0$, dans le cas contraire, il renvoie la valeur $b^\star = 1$.

L'heuristique qui guide cet algorithme est la suivante : si A a réussi à forger une empreinte du message m, c'est que B s'est comporté comme un véritable oracle au jeu de l'authentification, et les valeurs de la fonction fournies par l'oracle sont probablement celles d'un élément de la famille $\mathcal{F}$. Par contre, si l'adversaire A a échoué, on peut supposer que les valeurs fournies étaient celles d'une fonction parfaitement aléatoire.

Il reste à évaluer l'avantage de B. Celui-ci vaut :

$$\mathrm{Av}(B) = \Big|\underbrace{\Pr(b^\star = 0 \mid b = 0)}_{= \Pr_{\text{succes}}(A)} - \underbrace{\Pr(b^\star = 0 \mid b = 1)}_{= \Pr(E)}\Big|.$$

Si $b = 0$, alors l'algorithme B s'est comporté comme un véritable oracle auprès de A. Le premier terme ci-dessus est exactement la probabilité de succès de A. Le second terme est la probabilité d'un événement E qui énonce que « *A a réussi à forger une empreinte valide pour m alors que les réponses à ses requêtes étaient le résultat d'une fonction aléatoire.* » Évaluons la probabilité de l'événement E.

Soit F l'événement « *parmi les q requêtes formulées par l'algorithme A, il en existe deux, m et m' pour lesquelles les valeurs de la fonction f à l'avant dernière itération du CBC-MAC sont égales alors que les préfixes $m_1 \cdots m_{\ell-1}$ et $m'_1 \cdots m'_{\ell-1}$ sont différents.* » La fonction f de l'oracle étant aléatoire, la valeur à l'avant-dernière étape est aléatoire. Les préfixes des messages m et m' étant supposés différents, la probabilité de l'événement F est majorée par la probabilité de collision de deux valeurs aléatoires parmi les q requêtes. Cette dernière valeur est donnée par le paradoxe des anniversaire, d'où :

$$\Pr(F) \leq \frac{q^2}{2^{m+1}}.$$

En l'absence de collision, c'est-à-dire si l'événement F n'est pas survenu, la fonction f étant aléatoire, les réponses aux requêtes sont des valeurs aléatoires qui n'apportent aucune information. La probabilité que l'adversaire A trouve une empreinte valide ne dépasse pas la probabilité d'un tirage de sa valeur au hasard :

$$\Pr(E \mid \overline{F}) = \frac{1}{2^n}.$$

Par conséquent :

$$\Pr(E) = \underbrace{\Pr(E \mid F)}_{\leq 1} \times \underbrace{\Pr(F)}_{\approx \frac{q^2}{2^{n+1}}} + \underbrace{\Pr(E \mid \overline{F})}_{= \frac{1}{2^n}} \times \underbrace{\Pr(\overline{F})}_{\leq 1}$$

Ainsi, la probabilité $\Pr(E)$ est négligeable comme somme de deux quantités négligeables. L'avantage de B est la différence du succès de A qui est supposé non négligeable et d'un terme négligeable, donc $\mathrm{Av}(B)$ est non négligeable. □

Exercice 5.2. Montrer que si l'événement F défini dans la preuve ci-dessus survient sur deux messages m et m' connus, alors il est possible de forger une empreinte valide pour un message et de gagner au jeu de l'authentification.

La preuve ci-dessus suppose explicitement que les message à signer ont tous la même taille, et donc que la valeur de l'entier ℓ est fixe et détermine la taille de tous les messages. Le CBC-MAC tel que défini par la définition 1 ne convient donc pas pour des messages de taille variable.

Exercice 5.3. a) Montrer que si on autorise des messages de taille variable dans le CBC-MAC tel qu'il est donné par la définition 1, alors il est possible, grâce à un oracle, de forger une empreinte valide pour n'importe quel message de plus de deux blocs.

b) Modifier le CBC-MAC pour l'adapter à des messages de taille variable (proposer deux solutions).

4 Chiffrement IND-CCA générique

Outre l'authentification des messages, une application des codes d'authentification est la construction d'un système de confidentialité résistant à une attaque à cryptogrammes choisis. Plus précisément, la combinaison d'un système de chiffrement IND-CPA et d'un code d'authentification sûr conduit à un système de chiffrement IND-CCA, et ceci indépendamment du système de confidentialité choisi.

1. Considérons un système de confidentialité Γ dont l'espace des message est $\mathcal{M}$, l'espace des cryptogrammes est $\mathcal{E}$ et l'espace des clés est $\mathcal{K}$, et dont la fonction de chiffrement, indexée par la clé $k \in \mathcal{K}$ est :
$$E_k : \begin{array}{ccc} \mathcal{M} & \to & \mathcal{E} \\ m & \mapsto & E_k(m) \end{array}$$
On suppose que ce système de confidentialité a la propriété IND-CPA.
2. Considérons également un code d'authentification Σ, défini sur le même espace des messages $\mathcal{M}$, dont l'espace des codes est $\mathcal{S}$ et l'espace des clés est $\mathcal{K}'$, et dont la fonction de génération d'empreinte est, pour une clé $k' \in \mathcal{K}'$:
$$C_{k'} \begin{array}{ccc} \mathcal{M} & \to & \mathcal{S} \\ m & \mapsto & C_{k'}(m) \end{array}$$
On suppose que ce code d'authentification est sûr.

Associer ces deux primitives définit un système de confidentialité $\Gamma^\star$ comme suit :

1. L'espace des messages de $\Gamma^\star$ est l'espace des messages $\mathcal{M}$, commun à Γ et à Σ.
2. L'espace des clés est l'espace $\mathcal{K} \times \mathcal{K}'$ des couples (k, k') constitués d'une clé de chiffrement et d'une clé d'authentification. La fonction de génération de clé génère une clé k de chiffrement et une clé k' d'authentification.
3. L'espace des cryptogrammes est l'ensemble $\mathcal{E} \times \mathcal{S}$ des couples (c, σ) constitués du cryptogramme pour le système de confidentialité Γ et d'une empreinte pour le code d'authentification Σ.
4. La fonction de chiffrement $E^\star_{(k,k')}$ renvoie le couple (c, σ), constitué du cryptogramme $c = E_k(m)$ et l'empreinte $\sigma = C_{k'}(c)$ du cryptogramme c.
5. Pour déchiffrer le cryptogramme (c, σ), la fonction de déchiffrement $D'_{k,k'}$ procède comme suit :
   (a) Calculer la validité $v = V_{k'}(c, \sigma)$ de l'empreinte σ du cryptogramme c.
   (b) Si $v = 0$, alors renvoyer un symbole particulier $\perp$ pour signifier l'échec du déchiffrement.
   (c) Si $v = 1$, alors déchiffrer le cryptogramme c en utilisant la fonction de déchiffrement de Γ et renvoyer le message en clair $m = D_k(c)$.

L'adjonction d'une empreinte d'authentification de l'information chiffrée rend le cryptogramme non malléable. Modifier le cryptogramme entraine son invalidité et interdit d'appliquer la fonction de déchiffrement sur c. Le système de chiffrement ainsi obtenu résiste aux attaques à cryptogrammes choisis comme l'énonce le théorème suivant :

Théorème 4.1. Chiffrement IND-CCA générique

Avec les notations ci-dessus, si le système de chiffrement Γ est IND-CPA et si le code d'authentification Σ est sûr, alors le système de chiffrement $\Gamma^\star$ est IND-CCA.

Preuve Supposons que le chiffrement $\Gamma^\star$ n'est pas IND-CCA, c'est-à-dire qu'il existe un adversaire A, polynomial, contre le jeu de l'indistinguabilité avec accès à un oracle de déchiffrement, et qui a un avantage non négligeable. Montrons qu'alors le système de confidentialité Γ n'est pas IND-CPA ou bien le code d'authentification Σ n'est pas sûr. De manière équivalente, cela revient à montrer que si le code d'authentification Σ est sûr, alors le système de confidentialité Γ n'est pas indistinguable contre une attaque à clairs choisis.

Pour cela, construisons un adversaire B contre le jeu IND-CPA qui utilise A comme sous-programme. Cet algorithme joue au jeu de l'indistinguabilité contre un oracle et dispose d'un oracle de chiffrement. Il utilise également le code d'authentification Σ qui est supposé sûr. Il simule auprès de A l'oracle d'un jeu IND-CCA.

Algorithme 4.2. B **au jeu IND-CPA**

1. B génère une clé k' d'authentification pour le code d'authentification Σ.
2. B reçoit de A deux messages m_0 et m_1 qu'il transmet à l'oracle du jeu de l'indistinguabilité.
3. B reçoit de l'oracle le cryptogramme c_b du message m_b pour un choix aléatoire et inconnu de $b \in \{0, 1\}$.
4. B calcule l'empreinte de c_b et la joint à c_b. Il transmet le défi (c_b, τ) à A.
5. Lorsque l'algorithme A formule des requêtes de chiffrement, l'algorithme B transmet directement ces requêtes à l'oracle de chiffrement. Il reçoit un cryptogramme c. Il calcule l'empreinte τ de c et fournit le cryptogramme signé (c, τ) à A.
6. B mémorise tous les couples clairs-cryptogrammes obtenus au cours de ces requêtes dans une liste $\mathcal{L}$.
7. Lorsque A formule une requête de déchiffrement du cryptogramme (c, τ), l'algorithme B vérifie l'empreinte τ.
   - Si l'empreinte n'est pas correcte, il renvoie le symbole $\perp$ pour signifier l'échec du déchiffrement.
   - Si l'empreinte est correcte, il vérifie si le cryptogramme figure dans sa liste $\mathcal{L}$. Dans ce cas, il renvoie le message en clair correspondant.
   - Si le cryptogramme n'est pas dans la liste, il signifie l'échec du déchiffrement par l'envoi du symbole $\perp$.
8. L'algorithme B reçoit la réponse $b^\star$ de A qu'il renvoie lui-même comme valeur de retour.

Le succès de l'algorithme B est l'événement $\{b = b^\star\}$. Par définition, son avantage est :

$$\mathrm{Av}(B) = |2\Pr(b = b^\star) - 1|.$$

Soit E l'événement suivant : « *Il existe un cryptogramme valide parmi les requêtes de déchiffrement formulées par* A. » Si p est la probabilité que A formule une requête de déchiffrement valide, et si q est le nombre de telles requêtes formulées par A, la probabilité de l'événement $\overline{E}$ est :

$$\Pr(\overline{E}) = (1-p)^q \approx 1 - qp,$$

et donc :

$$\Pr(E) \approx qp \leq q\,\mathrm{Insec}(\Sigma),$$

où $\mathrm{Insec}(\Sigma)$ est l'insécurité du code d'authentification Σ, égale par définition à la borne supérieure de l'ensemble des probabilités de succès des adversaires. Le code d'authentification Σ étant supposé sûr, pour une valeur assez grande du paramètre de sécurité, on a par exemple $\Pr(\overline{E}) \geq 1/2$.

Décomposons la probabilité de succès de B selon l'occurrence ou non de l'événement E.

$$\begin{aligned}
\mathrm{Av}(B) &= \left|2\Pr(b = b^\star \mid E) \times \Pr(E) + 2\Pr(b = b^\star \mid \overline{E}) \times \Pr(\overline{E}) - 1\right| \\
&= \left|2\Pr(b = b^\star \mid E) \times \Pr(E) + 2\Pr(b = b^\star \mid \overline{E}) \times \Pr(\overline{E}) - P(E) - P(\overline{E})\right| \\
&\geq \left|\underbrace{\Pr(\overline{E})}_{\geq \frac{1}{2}} \underbrace{\left|2\Pr(b = b^\star \mid \overline{E}) - 1\right|}_{= \mathrm{Av}(A)} - \underbrace{\Pr(E)}_{\leq q\,\mathrm{Insec}(\Sigma)} \underbrace{\left|2\Pr(b = b^\star \mid E) - 1\right|}_{\leq 1}\right|
\end{aligned}$$

En effet, lorsque l'événement E ne survient pas, c'est-à-dire si aucune des requêtes de déchiffrement de A n'est valide, alors l'algorithme B se comporte auprès de A comme un véritable oracle de déchiffrement pour le jeu INC-CCA. Dans ce cas, le jeu est équitable pour A. La quantité $\Pr(b = b^\star \mid \overline{E})$ est exactement sa probabilité de succès au jeu IND-CPA, et $|2\Pr(b = b^\star \mid \overline{E}) - 1|$ est son avantage. On a finalement :

$$\mathrm{Av}(B) \geq \left|\frac{\mathrm{Av}(A)}{2} - q\,\mathrm{Insec}(\Sigma)\right|.$$

Le nombre de requêtes q étant majoré par un polynôme du paramètre de sécurité, l'avantage de A étant non négligeable et l'insécurité du code d'authentification étant négligeable, l'inégalité ci-dessus montre que l'avantage de B est supérieur à la différence d'un terme non négligeable et d'un terme négligeable. Il est donc non négligeable, et c'est ce qu'il fallait prouver. □

La preuve ci-dessus montre que le système de confidentialité qui associe un chiffrement sûr contre une attaque à clair choisi et un code d'authentification des cryptogrammes résiste non seulement à une attaque à cryptogramme choisis, mais plus généralement à une attaque à clairs et cryptogrammes choisis.

6 Chiffrement par blocs

Le chiffrement par blocs est la version moderne des substitutions alphabétiques utilisées en cryptographie traditionnelle. Ce procédé de chiffrement consiste à transformer un texte écrit en remplaçant chaque lettre par une autre. Le procédé le plus anciennement connu est celui de Caius Julius César, où chaque lettre est remplacée par une lettre située trois rangs plus loin dans l'alphabet. Un chiffrement par blocs moderne consiste à partager le message en blocs de 64 ou 128 symboles binaires et à faire opérer sur chacun d'eux une transformation inversible. Appliquer la transformation inverse permet de déchiffrer le cryptogramme. Ce procédé réalise finalement une substitution alphabétique sur un alphabet de taille considérable, égale à 2^{64} ou 2^{128} selon la taille du bloc.

Depuis que la cryptographie est entrée dans le domaine public, à partir des années 1970, ce type de procédé a fait l'objet de normes, tout d'abord avec le DES (*Data Encryption Standard*), premier algorithme cryptographique public pour l'usage civil en 1976, puis avec l'AES (*Advanced Encryption Standard*) en 2001. Plusieurs modes d'utilisation ont également été standardisés pour le chiffrement de messages de taille variable, citons principalement le mode dictionnaire (ECB, *Electronic CodeBook*), le mode chaîné (CBC *Cipher Block Chaining*) et le mode compteur (CTR *Counter*). À l'exception du mode compteur, ce type de chiffrement nécessite une famille de bijections indexée par une clé secrète. Ce chapitre est consacré à la construction et à l'évaluation de familles de fonctions inversibles qui ont les propriétés convenables pour réaliser un chiffrement par blocs.

1 Famille pseudo-aléatoire de permutations

Une famille pseudo-aléatoire de permutations n'est finalement rien d'autre qu'une famille pseudo-aléatoire de fonctions qui sont chacune des bijections un ensemble E.

Définition 1.1. Famille pseudo-aléatoire de permutations

Pour une valeur n du paramètre de sécurité, une famille échantillonnable $\mathcal{F} = (f_s)_{s\in\mathcal{F}}$ est une famille pseudo-aléatoire de permutations d'un ensemble E sur lui-même si les conditions suivantes sont réunies :

1. Chaque élément f_s est une bijection sur l'ensemble E, où $1/\mathrm{Card}(E)$ est une fonction négligeable.
2. Pour tout élément $x \in E$, et tout indice $s \in \mathcal{S}$, la valeur $f_s(x)$ est calculable par un algorithme polynomial.
3. Pour tout élément $y \in E$, et tout indice $x \in \mathcal{S}$, la valeur de l'antécédent $f_s^{-1}(y)$ est calculable par un algorithme polynomial.
4. La famille $\mathcal{F}$ est calculatoirement indistinguable de la famille $\mathcal{P}$ de toutes les permutations sur l'ensemble E.

Le dernier point caractérise le caractère pseudo-aléatoire de la famille. Il signifie que tout adversaire polynomial au jeu de la reconnaissance des familles ne peut distinguer un élément de la famille $\mathcal{F}$ d'une bijection aléatoire qu'avec un avantage négligeable.

L'adversaire a accès à un oracle de calcul des valeurs $f(x)$. Il existe une notion plus forte, dite de *famille super pseudo-aléatoire de permutations*, dont les éléments sont indistinguables d'une permutation aléatoire, y compris lorsque l'adversaire a accès à un oracle de calcul de la bijection inverse $y \mapsto f^{-1}(y)$.

Une famille pseudo-aléatoire de permutations définit un système de confidentialité, certes rudimentaire, mais sûr, comme le montre l'exercice qui suit.

Exercice 6.1. Soit $\mathcal{F} = (f_s)_{s\in\mathcal{S}}$ une famille pseudo-aléatoire de permutations sur l'ensemble E. On considère le système de confidentialité défini sur l'ensemble des messages $\mathcal{M} = E$ par la fonction de chiffrement suivante : pour un message $m \in E$ et une clé secrète $s \in \mathcal{S}$, le cryptogramme est donné par $c = f_s(m)$. Ce chiffrement étant déterministe, il n'est pas IND-CPA. Mais si l'on interdit à l'adversaire de requérir le chiffrement des messages m_0 et m_1 qu'il fournit en début de jeu, montrer que ce système de confidentialité est IND-CPA.

Le jeu honnête pour un adversaire contre une famille pseudo-aléatoire de permutations $\mathcal{F}$ consiste à lui faire reconnaître un élément de la famille $\mathcal{F}$ d'une permutation aléatoire. Mais en raison de l'injectivité, les valeurs d'une permutation aléatoire ne peuvent pas être considérées comme des tirages aléatoires et indépendants. Il est tentant de faire quand même jouer l'adversaire au jeu de la reconnaissance entre un élément de $\mathcal{F}$ et une fonction aléatoire non nécessairement bijective. Dans ce cas, le jeu n'est pas conforme et l'adversaire n'a pas le même avantage qu'au jeu pour lequel il est conçu. La proposition suivante montre que l'écart des deux avantages reste négligeable.

Proposition 1.2.

Soit A un adversaire au jeu de la reconnaissance entre un élément d'une famille $\mathcal{F}$ de permutation sur un ensemble E et un élément de la famille $\mathcal{P}$ de toutes les permutations sur E. Soit $\mathrm{Av}_{\mathcal{P}}(A)$ son avantage à ce jeu. Soit $\mathrm{Av}_{\mathcal{H}}(A)$ son avantage lorsqu'on le fait jouer au jeu de la reconnaissance entre un élément de $\mathcal{F}$ et un élément de la famille $\mathcal{H}$ de toutes les fonctions $E \to E$ non nécessairement bijectives.

L'écart $|\mathrm{Av}_{\mathcal{P}}(A) - \mathrm{Av}_{\mathcal{H}}(A)|$ est négligeable.

Preuve Lorsque A joue à la reconnaissance entre les familles $\mathcal{F}$ et $\mathcal{H}$, l'oracle choisit un symbole binaire b aléatoire dans $\{0,1\}$. Si $b = 0$, il tire au hasard un élément f de $\mathcal{F}$ et si $b = 1$, il tire au hasard un élément f de $\mathcal{H}$. L'adversaire A interroge l'oracle pour obtenir les valeurs $y = f(x)$ pour des paramètres x qu'il lui soumet. Il propose ensuite son estimation $b^\star$ de la valeur de b choisie par l'oracle. L'avantage de A à ce jeu est :

$$\mathrm{Av}_{\mathcal{H}}(A) = \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1)\Big|.$$

On peut supposer que les requêtes de A sont toutes distinctes. Lorsque $b = 1$, la différence entre une permutation aléatoire et une fonction aléatoire est qu'une permutation est toujours injective et les valeurs rendues par l'oracle dont toutes distinctes, alors que dans le cas d'une fonction, il est possible que deux éléments x et x' conduisent à des valeurs $f(x)$ et $f(x')$ égales.

Considérons l'événement $\mathcal{E}$ suivant :« *au cours des q requêtes, il existe deux valeurs y et y' rendues par l'oracle qui sont égales.* » Évaluons la probabilité $\Pr(b^\star = 0 \mid b = 1)$ selon que l'événement $\mathcal{E}$ est ou n'est pas survenu :

$$\Pr(b^\star = 0 \mid b = 1) = \underbrace{\Pr(b^\star = 0 \mid b = 1 \text{ et } \mathcal{E})}_{p_1} \times \Pr(\mathcal{E}) + \underbrace{\Pr(b^\star = 0 \mid b = 1 \text{ et } \overline{\mathcal{E}})}_{p_2} \times \Pr(\overline{\mathcal{E}})$$

La probabilité de l'événement $\mathcal{E}$ est donnée par le paradoxe des anniversaires :

$$\Pr(\mathcal{E}) \approx \frac{q^2}{2\mathrm{Card}(E)} \quad \text{et} \quad \Pr(\overline{\mathcal{E}}) \approx 1 - \frac{q^2}{2\mathrm{Card}(E)}$$

Cela conduit à :

$$\Pr(b^\star = 0 \mid b = 1) = \Pr(b^\star = 0 \mid b = 1 \text{ et } \overline{\mathcal{E}}) + \frac{q^2}{2\mathrm{Card}(E)} \underbrace{(p_1 - p_2)}_{-1 \le \eta \le 1}$$

L'avantage de A s'exprime donc :

$$\mathrm{Av}_{\mathcal{H}}(A) = \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1 \text{ et } \overline{\mathcal{E}}) + \eta\frac{q^2}{2\mathrm{Card}(E)}\Big|.$$

Lorsque $b = 1$ et lorsque l'événement $\mathcal{E}$ ne survient pas, l'adversaire A joue dans les conditions pour lesquelles il a été conçu. L'oracle aurait tout aussi bien pu choisir une permutation aléatoire. Son avantage est $\mathrm{Av}_{\mathcal{P}}(A)$. Donc :

$$\mathrm{Av}_{\mathcal{H}}(A) = \left|\mathrm{Av}_{\mathcal{P}}(A) + \eta\frac{q^2}{2\mathrm{Card}(E)}\right|.$$

Comme le réel η est compris entre -1 et 1, il en résulte la double inégalité suivante :

$$\mathrm{Av}_{\mathcal{P}}(A) - \frac{q^2}{2\mathrm{Card}(E)} \leq \mathrm{Av}_{\mathcal{H}}(A) \leq \mathrm{Av}_{\mathcal{P}}(A) + \frac{q^2}{2\mathrm{Card}(E)}.$$

Comme $1/\mathrm{Card}(E)$ est négligeable et q est borné par un polynôme du paramètre de sécurité, la quantité $q^2/2\mathrm{Card}(E)$ est négligeable, ce qui achève la preuve.

□

Exercice 6.2. Soit $\mathcal{F} = (f_s)_{s\in\{0,1\}^n}$ une famille pseudo-aléatoire de bijections sur l'ensemble $\{0,1\}^n$. Pour tout paramètre $s \in \{0,1\}^n$, on définit la fonction g_s par :

$$g_s : \begin{array}{ccc} \{0,1\}^{n+1} & \to & \{0,1\}^{n+1} \\ (x,b) & \to & \big(f_s(x), b \oplus x_0\big) \end{array}$$

La famille $\mathcal{G} = (g_s)_{s\in\{0,1\}^n}$ est-elle une famille pseudo-aléatoire de permutations ?

2 Schéma de Feistel

Cette section adresse la question de la construction de permutations pseudo-aléaoires. Le chapitre 3 permet de construire des familles pseudo-aléatoires de fonctions qui ne sont pas nécessairement inversibles. Une construction possible de bijections repose sur un schéma, appelé schéma de Feistel qui est un procédé de construction de permutation à partir de fonctions non nécessairement inversible. Il a été proposé par le cryptologue Horst Feistel(1915-1990) pour construire l'algorithme qui deviendra le DES.

Définition 2.1. Schéma de Feistel

Soit $(E,+)$ un groupe commutatif et f une application quelconque $E \to E$. On appelle schéma de Feistel la fonction :

$$\mathrm{Fe}_f : \begin{array}{ccc} E\times E & \to & E\times E \\ (x,y) & \mapsto & (y, x+f(y)) \end{array}$$

Un schéma de Feistel est toujours inversible. En notant Swp la fonction qui échange les composantes d'un vecteur de dimension 2, soit :

$$\mathrm{Swp} : \ (x,y) \mapsto (y,x),$$

il est immédiat de vérifier que la fonction $\mathrm{Swp} \circ \mathrm{Fe}_f$ est inversible et a pour inverse la fonction $\mathrm{Swp} \circ \mathrm{Fe}_f^-$, où Fe_f^- est définie par : $(x,y) \mapsto (y, x - f(y))$. Il en résulte la proposition suivante qui lie un schéma de Feistel et son inverse :

Proposition 2.2. Bijection inverse d'un schéma de Feistel

Soit f une application $E \to E$. Le schéma de Feistel Fe_f et son inverse Fe_f^{-1} sont reliés par la relation suivante :

$$\mathrm{Fe}_f^{-1} \circ \mathrm{Swp} = \mathrm{Swp} \circ \mathrm{Fe}_f^- .$$

La définition 1 ci-dessus un schéma de Feistel de manière général sur un groupe commutatif quelconque, mais en pratique il est utilisé principalement sur l'ensemble des mots binaires de longueur n muni de l'addition $\oplus$ qui est l'addition modulo 2 terme à terme. Dans ce cas, l'addition et la soustraction est la même opération et la fonction Fe_f^- est la même que la fonction Fe_f.

Notons $\mathcal{H}$ la famille de toutes les fonctions $E \to E$. On note $\mathrm{Fe}_{\mathcal{H}} = (\mathrm{Fe}_f)_{f\in\mathcal{H}}$ la famille des fonctions Fe_f lorsque f parcourt la famille $\mathcal{H}$. Comme la première composante de la valeur vaut la deuxième composante du paramètre, cette famille de permutation n'est pas pseudo-aléatoire. Il est en effet immédiat de construire un adversaire qui reconnaît à coup sûr cette particularité. Par conséquent, pour construire des familles pseudo-aléatoires de bijections, il sera nécessaire de composer ces schémas.

Définition 2.3. Composition des schémas de Feistel

Pour k fonctions $f_1, \dots f_k : E \to E$, on note $\mathrm{Fe}_{f_1\cdots f_k}$ la composition :

$$\mathrm{Fe}_{f_1\cdots f_k} = \mathrm{Fe}_{f_k} \circ \cdots \circ \mathrm{Fe}_{f_1}.$$

L'ordre des fonctions $f_1, \dots, f_k$ dans la notation $\mathrm{Fe}_{f_1\cdots f_k}$ est l'ordre dans lequel elles sont appliquées dans la composition, et non pas l'ordre dans lequel elles sont écrite dans la composition ci-dessus.

La proposition 2 ci-dessus se généralise directement :

Proposition 2.4.

Soient $f_1, \dots, f_k$ des fonctions $E \to E$. La composition des schémas de Feistel Fe_{f_i} et leurs inverses sont reliés par la relation :

$$(\mathrm{Fe}_{f_k} \circ \cdots \circ \mathrm{Fe}_{f_1})^{-1} \circ \mathrm{Swp} = \mathrm{Swp} \circ \mathrm{Fe}_{f_1}^- \circ \cdots \circ \mathrm{Fe}_{f_k}^-.$$

La relation donnée par la relation ci-dessus est très intéressante du point de vue pratique lorsqu'une fonction de chiffrement est définie par la composition de plusieurs schémas de Feistel et lorsque le groupe est $(\{0,1\}^n, \oplus)$. Dans ce cas, le déchiffrement est très exactement la fonction de chiffrement dans laquelle les fonctions f_i de chaque itération sont appliquées dans l'ordre inverse. L'algorithme DES exploite cette propriété.

Pour qu'un schéma de Feistel soit utilisable comme fonction de chiffrement, il est nécessaire de l'itérer plusieurs fois. Le principal résultat de cette section est une condition pour que la famille obtenue soit une famille pseudo-aléatoire de permutations :

- deux itérations sont insuffisantes,
- trois itérations suffisent lorsque les fonctions utilisées sur chaque tour appartiennent à une famille pseudo-aléatoire de fonctions.

Définition 2.5. Famille de permutations de Feistel

Soient $\mathcal{F}_1, \dots, \mathcal{F}_k$ des familles d'applications $E \to E$. Ces familles définissent par composition une famille de permutations par :

$$\mathrm{Fe}_{\mathcal{F}_1\cdots\mathcal{F}_k} = (\mathrm{Fe}_{f_1\cdots f_k})_{f_1\in\mathcal{F}_1,\dots,f_k\in\mathcal{F}_k}.$$

L'exercice suivant vise à démontrer que, même en prenant deux fonctions de tour f et g parfaitement

aléatoires, la composition de deux schémas de Feistel $\mathrm{Fe}_{fg} = \mathrm{Fe}_g \circ \mathrm{Fe}_f$ ne constitue pas une permutation pseudo-aléatoire.

Exercice 6.3. Soit $\mathcal{H}_E$ la famille de toutes les applications $E \to E$. Démontrer que la famille $\mathrm{Fe}_{\mathcal{H}\mathcal{H}}$ n'est pas une famille pseudo-aléatoire de permutations.

Enfin, le théorème suivant montre qu'à la condition de considérer une famille pseudo-aléatoire de fonctions, la composition de trois tours de schéma de Feistel conduit à une famille pseudo-aléatoire de permutations.

Théorème 2.6. Luby-Rackoff, 1988

Soit $\mathcal{F}$ une famille pseudo-aléatoire de fonctions, alors la famille $\mathrm{Fe}_{FFF} = (\mathrm{Fe}_{fgh})_{f,g,h\in\mathcal{F}}$ est une famille pseudo-aléatoire de permutations.

Preuve Il faut démontrer que la famille $\mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{F}}$ est indistinguable de la famille de toutes les permutations. D'après la proposition 2 page 62, il est équivalent de démontrer qu'elle est indistinguable de la famille $\mathcal{H}_{E\times E}$ de toutes les fonctions $E \times E \to E \times E$. La démonstration se fait en deux étapes.

La première étape consiste à démontrer que la famille $\mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{F}}$ des schémas de Feistel à trois tours dans lesquels les fonctions de tour sont des éléments de la famille pseudo-aléatoire $\mathcal{F}$ est indistinguable de la famille $\mathrm{Fe}_{\mathcal{H}_E\mathcal{H}_E\mathcal{H}_E}$ des schémas de Feistel à trois tours où les fonctions de tours sont aléatoires dans la famille $\mathcal{H}_E$ de toutes les fonctions $E \to E$. Elle est le résultat du lemme suivant.

Lemme 2.7.

Les trois inégalités suivantes sont satisfaites :

1. $\Delta(\mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{F}}, \mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{H}_E}) \leq \Delta(\mathcal{F}, \mathcal{H}_E)$
2. $\Delta(\mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{H}_E}, \mathrm{Fe}_{\mathcal{F}\mathcal{H}_E\mathcal{H}_E}) \leq \Delta(\mathcal{F}, \mathcal{H}_E)$
3. $\Delta(\mathrm{Fe}_{\mathcal{F}\mathcal{H}_E\mathcal{H}_E}, \mathrm{Fe}_{\mathcal{H}_E\mathcal{H}_E\mathcal{H}_E}) \leq \Delta(\mathcal{F}, \mathcal{H}_n)$

L'inégalité triangulaire sur les distances calculatoires des familles de fonctions et les trois inégalités du lemme ci-dessus montrent que :

$$\Delta(\mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{F}}, \mathrm{Fe}_{\mathcal{H}_E\mathcal{H}_E\mathcal{H}_E}) \leq 3\Delta(\mathcal{F}, \mathcal{H}_E).$$

Si la famille $\mathcal{F}$ est pseudo-aléatoire, c'est-à-dire si $\Delta(\mathcal{F}, \mathcal{H}_E)$ est négligeable, alors le second membre de l'inégalité ci-dessus l'est, et le premier membre aussi.

On en déduit que la distance calculatoire $\Delta(\mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{F}}, \mathrm{Fe}_{\mathcal{H}_E\mathcal{H}_E\mathcal{H}_E})$ est négligeable.

La seconde étape consiste à démontrer que la famille $\mathrm{Fe}_{\mathcal{H}_E\mathcal{H}_E\mathcal{H}_E}$ est elle-même indistinguable de la famille $\mathcal{H}_{E\times E}$, comme l'énonce le lemme suivant :

Lemme 2.8.

Si les trois fonctions de tour d'un schéma de Feistel à trois tours sont aléatoires, alors ce schéma est indistinguable d'une bijection aléatoire.

Ce lemme montre que la distance calculatoire $\Delta(\mathrm{Fe}_{\mathcal{H}_E\mathcal{H}_E\mathcal{H}_E}, \mathcal{H}_{E\times E})$ est négligeable. L'application de l'inégalité triangulaire permettra alors de conclure que la distance calculatoire $\Delta(\mathrm{Fe}_{\mathcal{F}\mathcal{F}\mathcal{F}}, \mathcal{H}_{E\times E})$ est elle-même négligeable, car inférieure à la somme de deux quantités négligeables, ce qui achève la preuve. □

Il ne reste plus qu'à démontrer les deux lemmes sur lesquels repose la preuve du théorème de Luby-Rackoff.

Preuve du lemme 7 Montrons le premier point du lemme, les deux autres se démontrent de la même façon et sont laissés en exercice.

Soit A un adversaire au jeu de l'indistinguabilité des familles $\mathrm{Fe}_{\mathcal{FFF}}$ et $\mathrm{Fe}_{\mathcal{FF}\mathcal{H}_E}$. Construisons un adversaire B_A au jeu de l'indistinguabilité des familles $\mathcal{F}$ et $\mathcal{H}_E$. Cet adversaire joue contre un oracle qui a choisi un symbole binaire aléatoire $b \in_R \{0,1\}$. Si $b = 0$, il réalise un élément aléatoire de la famille $\mathcal{F}$ et si $b = 1$ il réalise une fonction aléatoire de $\mathcal{H}_E$. L'algorithme B_A utilise l'algorithme A comme sous-programme auprès de qui il simule un oracle. Soit B_A l'algorithme suivant :

Algorithme 2.9. B_A

1. B_A choisit au hasard deux fonctions f_1 et f_2 dans $\mathcal{F}$.
2. Lorsque A sollicite la valeur de l'élément (x, y) de $E \times E$, l'algorithme B_A calcule la valeur du schéma de Feistel à trois tour de la façon suivante :
   — Il utilise f_1 et f_2 pour les deux premiers tours ;
   — Il sollicite l'oracle pour la fonction du troisième tour.
3. Lorsque A a terminé, l'algorithme B_A renvoie la même valeur $b^\star$ que celle rendue par A.

Par construction, le succès de l'algorithme B_A est exactement le succès de l'algorithme A. Les deux algorithmes ont par conséquent le même avantage :

$$\mathrm{Av}(B_A) = \mathrm{Av}(A).$$

Soit $\mathcal{A}$ l'ensemble de tous les distingueurs de $\mathrm{Fe}_{\mathcal{FFF}}$ et $\mathrm{Fe}_{\mathcal{FF}\mathcal{H}_E}$. Soit $\mathcal{B}$ l'ensemble de tous les distingueurs de $\mathcal{F}$ et $\mathcal{H}_E$. Par passage à la borne supérieure, on a :

$$\underbrace{\sup_{A\in\mathcal{A}} \mathrm{Av}(A)}_{=\,\Delta(\mathrm{Fe}_{\mathcal{FFF}},\,\mathrm{Fe}_{\mathcal{FF}\mathcal{H}_E})} = \sup_{A\in\mathcal{A}} \mathrm{Av}(B_A) \leq \underbrace{\sup_{B\in\mathcal{B}} \mathrm{Av}(B)}_{=\,\Delta(\mathcal{F},\,\mathcal{H}_E)}.$$

□

Exercice 6.4. Démontrer de la même façon le points 2. et 3. du lemme 7.

Preuve du lemme 8 On considère un schéma de Feistel à trois tours dont les fonctions de tours sont respectivement f, g et h. Pour une entrée $(x_0, y_0) \in E \times E$, on note $(x_1, y_1) = (y_0, x_0 + f(y_0))$, $(x_2, y_2) = (y_1, x_1 + g(y_1))$ et $(x_3, y_3) = (y_2, x_2 + h(x_2))$ les valeurs intermédiaires à chaque tour.

Soit A un adversaire au jeu de l'indistinguabilité. Montrons que l'avantage de A reste négligeable. L'expression de l'avantage de A est donnée par :

$$\mathrm{Av}(A) = \Big|\Pr(b^\star = 0 \mid b = 0) - \Pr(b^\star = 0 \mid b = 1)\Big|,$$

où b est le choix de l'oracle et $b^\star$ la réponse de A. Soit m le nombre de requêtes effectuées par l'adversaire A.

Si l'adversaire A formule deux fois la même requête, il obtiendra toujours la même réponse. On peut donc supposer sans restriction que les requêtes de A sont toutes deux à deux distinctes. Dans le cas contraire, il est immédiat de construire un adversaire équivalent qui satisfait cette condition.

Si au cours des m requêtes de l'algorithme A, les entrées des fonctions g et h du schéma de Feistel sont deux-à-deux distinctes, alors, les fonctions g et h étant aléatoires, les valeurs x_3 et y_3 sont aléatoires. Rien ne distingue alors la valeur d'un schéma de Feistel d'une fonction réellement aléatoire.

Soit $\mathcal{E}_m$ l'événement « *Il existe deux requêtes distinctes $i < j \leq m$ pour lesquelles les entrées y_1^i et y_1^j de la fonction g, ou les entrées y_2^i et y_2^j de la fonction h sont égales.* »

Comme les fonctions g et h sont aléatoires, la situation où $b = 0$ et où l'événement $\mathcal{E}_m$ n'est pas survenu provoque des sorties aléatoires et n'est donc pas distinguable de la situation $b = 1$. Il en résulte l'égalité :

$$\Pr(b^\star = 0 \mid b = 1) = \Pr(b^\star = 0 \mid b = 0 \text{ et } \overline{\mathcal{E}_m}).$$

En conditionnant selon que l'événement $\mathcal{E}_m$ est ou n'est pas survenu, on a :

$$\begin{aligned}\Pr(b^\star = 0 \mid b = 0) &= \Pr(b^\star = 0 \mid b = 0 \text{ et } \mathcal{E}_m) \times \Pr(\mathcal{E}_m) + \Pr(b^\star = 0 \mid b = 0 \text{ et } \overline{\mathcal{E}_m}) \times (1 - \Pr(\mathcal{E}_m)) \\ &= \Pr(E_m)\Big(\underbrace{\Pr(b^\star = 0 \mid b = 0 \text{ et } \mathcal{E}_m) - \Pr(b^\star = 0 \mid b = 0 \text{ et } \overline{\mathcal{E}_m})}_{\text{compris entre } -1 \text{ et } +1}\Big) + \Pr(b^\star = 0 \mid b = 1)\end{aligned}$$

Il en résulte que :

$$\mathrm{Av}(A) \leq \Pr(\mathcal{E}_m).$$

Il reste à démontrer que la probabilité de l'événement $\mathcal{E}_m$ est négligeable. Montrons qu'on a :

$$\Pr(\mathcal{E}_m) \leq \frac{m^2}{\mathrm{Card}(E)},$$

ce qui achèvera la preuve, car m est majoré par un polynôme en le paramètre de sécurité.

Pour tout entier k compris entre 2 et m, l'événement $\mathcal{E}_k$ est la réunion disjointe de $\mathcal{E}_k \cap \mathcal{E}_{k-1}$ et de $\mathcal{E}_k \cap \overline{\mathcal{E}_{k_1}}$. De plus, on a l'inclusion $\mathcal{E}_k \subset \mathcal{E}_{k-1}$. D'où :

$$\Pr(\mathcal{E}_k) = \Pr(\mathcal{E}_{k-1}) + \Pr(\mathcal{E}_k \mid \overline{\mathcal{E}_{k-1}}) \times \Pr(\overline{\mathcal{E}_{k-1}}) \leq \Pr(\mathcal{E}_{k-1}) + \Pr(\mathcal{E}_k \mid \overline{\mathcal{E}_{k-1}}).$$

Il en résulte que :

$$\Pr(\mathcal{E}_m) \leq \sum_{k=2}^{m} \Pr(\mathcal{E}_k \mid \overline{\mathcal{E}}_{k-1}).$$

Si l'événement $\mathcal{E}_k$ survient, mais pas l'événement $\mathcal{E}_{k-1}$, c'est que l'égalité des entrées a lieu aux requêtes i et k. Il existe donc une requête i strictement antérieure à k pour laquelle l'entrée de g ou de h est égale respectivement à l'entrée de g ou de h à la requête k. Donc :

$$\Pr(\mathcal{E}_k \mid \overline{\mathcal{E}_{k-1}}) \leq \sum_{i=1}^{k-1} \Pr(y_1^i = y_1^k) + \sum_{i=1}^{k-1} \Pr(y_2^i = y_2^k).$$

Si $y_0^i \neq y_0^k$, comme la fonction f est aléatoire, l'événement $\{y_1^i = y_1^k\}$ survient avec une probabilité $1/\mathrm{Card}(E)$.

Si $y_0^i = y_0^k$, alors, comme on peut supposer les requêtes différentes, on a $x_0^i \neq x_0^k$, donc $y_1^i \neq y_1^k$.

On en déduit que $\Pr(y_1^i = y_1^k) \leq 1/\mathrm{Card}(E)$, donc $\sum_{i=1}^{k-1} \Pr(y_1^i = y_1^k) \leq (k-1)/\mathrm{Card}(E)$.

On a un résultat similaire en remplaçant l'indice 1 par l'indice 2, du fait que la fonction g également est aléatoire. Donc :

$$\Pr(\mathcal{E}_k \mid \overline{\mathcal{E}_{k-1}}) \leq \frac{k-1}{\mathrm{Card}(E)} + \frac{k-1}{\mathrm{Card}(E)} = \frac{2(k-1)}{\mathrm{Card}(E)}$$

D'où finalement :

$$\Pr(\mathcal{E}_m) \leq \sum_{k=1}^{m-1} \frac{2k}{\mathrm{Card}(E)} \leq \frac{m(m-1)}{\mathrm{Card}(E)} \leq \frac{m^2}{\mathrm{Card}(E)},$$

qui est négligeable. □

Exercice 6.5. Considérons la famille des schémas de Feistel à trois tours dans laquelle la fonction centrale f_2 est définie par $f_2 : \ y \mapsto y \oplus i(y)$ où la fonction i est involutive, c'est-à-dire $i \circ i$ est l'identité.

1. Montrer que $f_2 \circ i = f_2$.
2. En déduire alors que la famille de fonctions obtenue n'est plus pseudo-aléatoire.

   Indication : Trouver deux requêtes différentes pour lesquelles les premières composantes de la valeur sont égales.

3 Chiffrement chaîné CBC

Une primitive de calcul par bloc peut être utilisée pour le chiffrement de messages de taille fixe, égale à la taille du bloc. Plusieurs modes d'utilisation font l'objet de normes de manière à pouvoir utiliser une telle primitive pour chiffrer des messages de taille variable, égale à un multiple de la taille du bloc. Deux modes ont déjà été décrits.

— Le mode ECB n'est pas sûr (voir l'exercice 4, page 48).
— Le mode compteur (mode CTR) est IND-CPA (voir exercice 6, page 51).

L'objet de ce paragraphe est de décrire le mode chaîné (CBC *Cipher Block Chaining*) et de prouver sa sécurité IND-CPA.

Dans le mode de chiffrement chaîné, le chiffrement des blocs ne sont pas indépendants les uns des autres. Le chiffrement du i-ième bloc de message dépend du cryptogramme du bloc précédent. Cela nécessite un vecteur initial supplémentaire qui constitue le premier bloc du cryptogramme.

Définition 3.1. Chiffrement CBC

Soit $\mathcal{F} = (f_s)_{s\in\mathcal{S}}$ une famille pseudo-aléatoire de permutations sur $\{0,1\}^n$. Le chiffrement CBC est le système de confidentialité défini par les éléments suivants :

— L'ensemble $\mathcal{M}$ des messages est l'ensemble des séquences finies $m = m_1 \cdots m_\ell$ d'éléments de $\{0,1\}^n$, dont le nombre ℓ est majoré par un polynôme en n.
— L'ensemble $\mathcal{C}$ des cryptogrammes est l'ensemble des séquences finies $c_0 \cdots c_\ell$ d'éléments de $\{0,1\}^n$.
— L'ensemble des clés est l'ensemble $\mathcal{S}$ des indices de la famille $\mathcal{F}$.
— La fonction de chiffrement du message $m = m_1 \cdots m_\ell$ avec la clé $s \in \mathcal{S}$ est la suivante :

1. Générer un vecteur initial c_0 aléatoire dans $\{0,1\}^n$.
2. Pour $i = 1$ à ℓ, poser $c_i = f_s(m_i \oplus c_{i-1})$.
3. Le cryptogramme est $c = c_0 c_1 \ldots c_\ell$.

Exercice 6.6. Écrire la fonction de déchiffrement.

Exercice 6.7. On suppose que lors de la transmission, le bloc de cryptogramme c_k a été corrompu ou perdu. Quels sont les blocs du message en clair qui seront affectés après déchiffrement ?

Exercice 6.8. Démontrer que le chiffrement CBC n'est pas IND-CCA.

Exercice 6.9. Démontrer qu'un chiffrement CBC avec un vecteur initial prévisible n'est pas IND-CPA.

Le principal résultat de ce paragraphe est la sécurité chiffrement CBC face à une attaque à clairs choisis.

Pour affirmer la sécurité du chiffrement CBC, il est nécessaire que le vecteur initial soit aléatoire ou du moins imprévisible. Comme le montre l'exercice 9 ci-dessus, un vecteur initial, même différent pour chaque message, comme par exemple un compteur, lorsqu'il est prévisible, n'assure pas la sécurité du chiffrement CBC.

Théorème 3.2. Sécurité du chiffrement CBC

Si la famille de permutations $\mathcal{F} = (f_s)_{s\in\mathcal{S}}$ est pseudo-aléatoire, alors le chiffrement CBC est IND-CPA.

Preuve Supposons l'existence d'un adversaire A au jeu de l'indistinguabilité avec accès à un oracle de chiffrement et dont l'avantage est non négligeable. À partir de cet adversaire, construisons un adversaire B qui distingue la famille $\mathcal{F}$ de la famille de toutes les permutations et qui a aussi un avantage non négligeable. Ce résultat est en contradiction avec l'hypothèse selon laquelle la famille $\mathcal{F}$ est pseudo-aléatoire. L'algorithme B joue contre un oracle qui a choisi un symbole binaire $\widehat{b} \in_R \{0,1\}$. Si $\widehat{b} = 0$, l'oracle choisit une fonctions $f = f_s$ aléatoire dans $\mathcal{F}$ et si $\widehat{b} = 1$, il choisit une fonction f aléatoire dans l'ensemble de toutes les fonctions sur $\{0,1\}^n$. L'algorithme B doit estimer la valeur de $\widehat{b}$ et pour cela, va utiliser l'adversaire A comme sous-programme et simuler auprès de lui l'oracle du jeu de l'indistinguabilité.

Algorithme 3.3. B

1. B reçoit de A deux messages m_0 et m_1.
2. B choisit un symbole binaire aléatoire $b \in_R \{0,1\}$ et chiffre le message m_b.
3. Pour chiffrer les messages, B fait appel à l'oracle avec la valeur x pour obtenir l'image $f(x)$ et calculer le cryptogramme.
4. L'algorithme A renvoie sa réponse $b^\star$. Si $b^\star = b$, alors B renvoie $\widetilde{b} = 0$ et si $b^\star \neq b$, alors B renvoie la valeur $\widetilde{b} = 1$.

L'heuristique qui guide cet algorithme est que, si A a trouvé la bonne valeur de b, c'est que le jeu était probablement honnête pour lui, et que les cryptogrammes ont été construits à partir d'une fonction pseudo-aléatoire, alors que si A a perdu, il a probablement été tenu en échec du fait que les cryptogrammes étaient construits à partir d'une fonction aléatoire.

Il reste à évaluer l'avantage de cet algorithme B et montrer qu'il est non négligeable.

$$\begin{aligned}
\mathrm{Av}(B) &= \left|\Pr(\widetilde{b}=0 \mid \widehat{b}=0) - \Pr(\widetilde{b}=0 \mid \widehat{b}=1)\right| \\
&= \left|\Pr(b^\star = b \mid \widehat{b}=0) - \Pr(b^\star = b \mid \widehat{b}=1)\right| \\
&\geq \Big|\underbrace{\left|\Pr(b^\star = b \mid \widehat{b}=0) - \frac{1}{2}\right|}_{=\frac{\mathrm{Av}(A)}{2}} - \left|\Pr(b^\star = b \mid \widehat{b}=1) - \frac{1}{2}\right|\Big|
\end{aligned}$$

Lorsque $\widehat{b} = 0$, le jeu est honnête pour A et l'événement $\{b^\star = b\}$ représente son succès.

Lorsque $\widehat{b} = 1$, les cryptogrammes sont construits avec une fonction aléatoire. Considérons l'événement E suivant : « *les valeurs demandées à l'oracle pour le chiffrement du message m_b sont toutes différentes de celles demandées à l'oracle pour le chiffrement de ses requêtes* ». Lorsque l'événement E est réalisé, le cryptogramme c_b est purement aléatoire, il a autant de chance d'être celui de m_0 que celui de m_1. Conditionnons la probabilité de succès de A dans ce cas selon que l'événement E est ou n'est pas survenu.

$$\begin{aligned}
\Pr(b = b^\star \mid \widehat{b}=1) &= \underbrace{\Pr(b = b^\star \mid \widehat{b}=1 \text{ et } E)}_{=\frac{1}{2}}\underbrace{\Pr(E)}_{1-\Pr(\overline{E})} + \underbrace{\Pr(b = b^\star \mid \widehat{b}=1 \text{ et } \overline{E})}_{0\leq\eta\leq 1}\Pr(\overline{E}) \\
\Pr(b = b^\star \mid \widehat{b}=1) - \frac{1}{2} &= \Pr(\overline{E})\underbrace{\left(\eta - \frac{1}{2}\right)}_{\text{compris entre } -1/2 \text{ et } 1/2}
\end{aligned}$$

$$\left|\Pr(b = b^\star \mid \widehat{b} = 1) - \frac{1}{2}\right| \quad \leq \quad \frac{\Pr(\overline{E})}{2}$$

La fin de la preuve résulte directement du lemme suivant qui énonce que la probabilité de l'événement $\overline{E}$ est négligeable, et l'avantage de B est non négligeable comme supérieur à la différence d'un terme non négligeable et d'un terme négligeable.

□

Lemme 3.4.

Si l'algorithme A formule q requêtes de moins de ℓ blocs chacune, alors :

$$\Pr(\overline{E}) \leq \frac{q\ell^2}{2^n}$$

Preuve Les requêtes sont aléatoires, car elles sont le résultat de la somme d'un bloc de message et d'un bloc de cryptogramme précédent qui est la valeur d'une fonction aléatoire. La probabilité qu'une demande à l'oracle ne soit pas l'une des au plus ℓ formulées pour le chiffrement du défi est donc supérieure ou égale à $1 - \frac{\ell}{2^n}$.

La probabilité qu'aucune ne le soit parmi les $q \times \ell$ demande à l'oracle pour le chiffrement des q requêtes de A, est la probabilité de l'événement E :

$$\Pr(E) \geq \left(1 - \frac{\ell}{2^n}\right)^{q\ell} \approx 1 - \frac{q\ell^2}{2^n}.$$

Le résultat du lemme s'en déduit immédiatement. □

Exercice 6.10. Soit CBC* le mode de chiffrement similaire au mode CBC mais dans lequel on échange les procédures de chiffrement et de déchiffrement. Ce mode reste-t-il IND-CPA ?

7
Signatures numériques

La signature numérique est un service qui contribue pleinement à la confiance des acteurs dans le monde des communications électroniques. Cette fonctionnalité a sans aucun doute accéléré son développement. Une signature non contestable et au cœur de nombreuses relations contractuelles et sa nature profondément asymétrique – une clé privée pour signer un document et une clé publique pour vérifier la signature – l'ancre dans la cryptographie à clé publique.

Une signature asymétrique est la version numérique du tableau d'informations municipales adossé au bâtiment des mairies. Tout passant peut consulter ces informations, mais seul l'employé municipal qui dispose de la clé d'accès peut y apposer les annonces.

La signature numérique associe deux fonctions distinctes :

— Le calcul de la signature fait intervenir le document à signer et une clé privé détenue par l'autorité habilitée à délivrer la signature ;

— La fonction de vérification invoque une clé publique et permet à tous de s'assurer que la signature adossée au document a bien été produite par l'autorité annoncée.

La signature numérique assure finalement les services suivants :

— Le service d'**intégrité** : toute modification du document invalide la signature.

— Le service d'**authentification** : une signature vérifiée assure qu'elle a bien été produite par le détenteur de la clé privée qui correspond à la clé publique utilisée pour la vérification.

— Le service de **non-répudiation** : la signature engage le signataire. Elle ne peut être produite par personne d'autre que le détenteur de la clé privée. Le signataire ne peut nier avoir signé le document.

La signature numérique est l'un des nouveaux mécanismes décrits dans l'article fondateur de la cryptographie à clé publique de Whitfield Diffie (né en 1944) et Martin Hellman (né en 1945) *New Directions in Cryptography*, publié en 1976, [3]. Cet article a marqué une frontière entre la cryptographie traditionnelle, où chaque acteur doit disposer d'un secret identiquement partagé avec son correspondant pour communiquer en toute confiance, et la nouvelle cryptographie qui repose sur des clés asymétriques aux rôles distincts.

Les premiers systèmes de signature à clé publique ont été définis à partir d'une fonction à sens unique avec trappe, c'est-à-dire une information tenue secrète qui permet à son détenteur de trouver un antécédent. Un apports de la théorie cryptographique est d'avoir établi que, contrairement au chiffrement à clé publique, un mécanisme de signature n'a pas besoin de trappe. Il peut être élaboré à partir d'une simple fonction à sens unique. Ce résultat paradoxal montre que la frontière des mécanisme cryptographiques ne repose pas tant sur la nature – asymétriques ou non – des clés utilisées, mais sur la nature des primitives sur lesquels s'appuie leur définition. En ce sens, la signature numérique appartient au même monde que le chiffrement symétrique, le chiffrement asymétrique appartenant, lui, à un autre monde.

Ce chapitre définit la sécurité des systèmes de signature et propose plusieurs constructions.

1 Définition et sécurité des signatures numériques

Dans son principe, la définition d'une signature numérique est celle d'un système d'authentification à clé publique. La clé pour produire la signature d'un document est différente de celle qui permet de vérifier la signature. De plus, la clé privée de signature ne se déduit pas aisément de la clé publique de vérification. La définition formelle d'un système de signature numérique est l'association de trois ensembles et de trois algorithmes.

Définition 1.1. Système de signature asymétrique

Un système de signature est la donnée des éléments suivants :

- Un ensemble $\mathcal{M}$ de messages.
- Un ensemble $\mathcal{S}$ de signatures.
- Un ensemble $\mathcal{K} = \mathcal{K}_{\text{pub}} \times \mathcal{K}_{\text{priv}}$ de couples de clés $(k_{\text{pub}}, k_{\text{priv}})$.
- Un algorithme efficace Gen qui génère un couple aléatoire de clés dans $\mathcal{K}$.
- Une fonction efficace de production de signature : $\text{Sign} : \mathcal{M} \times \mathcal{K}_{\text{priv}} \to \mathcal{S}$, qui permet de signer un message avec une clé privée.
- Une fonction efficace de vérification de signature : $\text{Verif} : \mathcal{M} \times \mathcal{S} \times \mathcal{K}_{\text{pub}} \to \{0, 1\}$ qui, à partir d'un message, sa signature et une clé publique calcule une valeur, égale à 1 si la signature est valide et égale à 0 si la signature ne l'est pas.

La fonction de génération Sign et la fonction de vérification Verif satisfont la condition de validité suivante :

$$\forall (k_{\text{pub}}, k_{\text{priv}}) \in \mathcal{K},\ \forall m \in \mathcal{M},\ \text{Verif}(m, \text{Sign}(m, k_{\text{priv}}), k_{\text{pub}}) = 1.$$

La condition de validité exprime que la signature d'un message correctement signé est toujours considérée comme valide par l'algorithme de vérification.

La clé privée ne doit pas pouvoir se déduire de la clé publique. Comme les clés sont distinctes, la procédure de vérification générique (définition 2, page 54) applicable aux MAC n'est pas applicable aux systèmes de signature asymétriques.

Comme pour les codes d'authentification, la sécurité d'un système de signature asymétrique se mesure à la difficulté de forger des fausses signatures en l'absence de la connaissance de la clé privée. Cette difficulté se mesure à l'avantage d'un adversaire au jeu de l'authentification. Ce jeu ne diffère de celui pour les MAC (jeu 1 page 55) que par le fait que les clés de génération et de vérification sont distinctes et que cette dernière est connue de l'adversaire.

Le jeu de l'authentification oppose un adversaire à un oracle.

Jeu 1.2. de l'authentification

1. L'oracle génère un couple de clé $(k_{\text{pub}}, k_{\text{priv}})$ dans $\mathcal{K}$ et transmet la clé publique k_{pub} à l'adversaire.
2. L'adversaire soumet à l'oracle des messages m_i, ce dernier lui renvoie la signature du message $\sigma_i = \text{Sign}(m, k_{\text{priv}})$.
3. Au bout d'un temps polynomial, l'adversaire fournit un message signé $(m, \sigma) \in \mathcal{M} \times \mathcal{S}$, où m est différent des m_i soumis à l'étape 2.
4. L'adversaire a gagné si la signature passe la fonction de vérification, c'est-à-dire si $\text{Verif}(m, \sigma, k_{\text{pub}}) = 1$. Il a perdu dans le cas contraire.

La performance d'un adversaire se mesure à sa probabilité de gagner au jeu de l'authentification.

Le jeu 2 ci-dessus place l'adversaire dans les conditions les plus favorables, c'est-à-dire les conditions sont les plus sévères pour la sécurité. Il peut faire signer des messages de son choix par l'oracle pour se faire aider. Il s'agit de l'attaque à message choisis, notée UF-CMA (*Unforgeable under Chosen Messages Attack*).

Il existe des conditions plus faibles de sécurité qui sont moins favorables à l'adversaire :

- L'observation passive : l'adversaire peut accéder à un ensemble de messages correctement signés connus, mais non choisis. Cette attaque est la plus vraisemblable dans une situation réelle.
- Pas d'observation : l'adversaire ne peut accéder à aucun message signé. Cette condition est considérée comme trop faible. En pratique, il est toujours possible d'observer des messages signés.

Présentons quelques systèmes classiques de signature :

La signature RSA

Le premier algorithme de signature à avoir été proposé est la signature RSA qui a été publié en 1978 par Rivest, Shamir et Adleman [[10]. La clé publique est un entier n, égal au produit de deux nombres premiers distincts p et q tenus secrets, associé à un exposant public e, premier avec $p-1$ et $q-1$. La clé privée est l'exposant privé d, égal à l'inverse de e modulo le ppcm de $p-1$ et $q-1$. L'ensemble des messages et l'ensemble des signatures est l'ensemble $\mathbb{Z}/n\mathbb{Z}$. La signature du message m est $\sigma = m^d \bmod n$. Pour vérifier la signature σ du message m, on calcule $\mu = \sigma^e \bmod n$ et on vérifie que $\mu = m$.

Ce mécanisme élémentaire n'a pas la sécurité requise, comme le montre l'exercice qui suit.

Exercice 7.1. a) Démontrer que la signature RSA n'est pas sûre, y compris dans le modèle le moins exigeant, où l'adversaire ne connaît que les paramètres publics et n'a accès à aucun document signé.
b) Démontrer que, si on interdit les messages dont la signature est triviale, la signature RSA n'est pas sûre contre une attaque à messages connus.

Le paradigme *hache-puis-signe*

Pour sécuriser le système de signature RSA, une solution est d'appliquer le paradigme *hache-puis-signe* qui consiste à soumettre tout d'abord le message à une fonction de hachage. Cette fonction de hachage produit une empreinte du message. La fonction de signature est ensuite appliqué à l'empreinte produite.

Soit $H : \mathcal{M} \to \mathcal{E}$ une fonction de hachage qui produit une empreinte h pour un message m. Dans le paradigme *hache-puis-signe*, la production et la vérification de signature se calculent ainsi :

$$\begin{aligned} \sigma &= \mathrm{Sign}(H(m), k_{\mathrm{priv}}) \\ v &= \mathrm{Verif}(H(m), \sigma, k_{\mathrm{pub}}) \end{aligned}$$

Une fonction de hachage idéale est une fonction aléatoire publiquement accessible via un oracle qui renvoie la valeur lorsqu'on lui soumet un message. Un tel modèle s'appelle un *oracle aléatoire*.

Encadré 1.3. Oracle aléatoire

Un oracle aléatoire est le modèle de fonction de hachage idéale. Il s'agit par définition d'une fonction H aléatoire de l'ensemble $\mathcal{M}$ des messages vers un ensemble fini E d'empreintes.

Pour tout message $m \in \mathcal{M}$, la valeur $H(m)$ est aléatoire et indépendante des valeurs pour les autres messages.

Dans ce modèle, la valeur $H(m)$ n'est pas calculée par un algorithme public, mais est demandée à un oracle qui tire un élément aléatoire dans E à chaque fois qu'il est sollicité pour un nouveau message. La fonction de signature ainsi que la fonction de vérification doivent faire appel à l'oracle pour leur calcul.

Les valeurs sont mémorisées de telle sorte que l'empreinte soit une fonction des messages et que l'oracle rende la même valeur pour toutes les sollicitations avec le même message.

Les réponses de l'oracle sont publiques et connues de tous. Ainsi, tout algorithme peut interroger l'oracle pour le message de son choix, eti, pour un message donné, l'oracle fournit toujours la même réponse à tous.

Un oracle aléatoire est un modèle théorique qui n'a pas de réalité pratique. Ce modèle est commode pour établir les preuves de sécurité des systèmes de signature ou de chiffrement, mais il y a un débat sur la pertinence de ce modèle dans le monde réel. Les fonctions de hachage utilisées en pratique (SHA1, MD5, *etc.*) sont loin d'être conformes à ce modèle, et la théorie cryptologique s'efforce de concevoir des mécanismes de protection qui ne font pas appel à un oracle aléatoire.

Le modèle de l'oracle aléatoire consacre l'utilisation de deux fonctions distinctes matérialisées par deux algorithmes indépendants pour signer un message : l'un hache le message pour définir une empreinte du message et l'autre signe l'empreinte pour produire la signature. Il en résulte que les algorithmes de signature d'un message et de vérification de la signature sont dotés d'une interface pour appeler l'oracle et requérir l'empreinte du message.

Proposition 1.4. Sécurité de l'association oracle aléatoire et signature RSA

Soit m un module RSA, soient e et d des exposants publics et privés. On considère le mécanisme de signature où la signature d'un message $m \in \mathcal{M}$ est $\sigma = H(m)^d$. Si la fonction RSA_e est à sens unique en l'absence de la connaissance de d et si H est un oracle aléatoire, alors ce système de signature est sûr contre une attaque à messages choisis.

Preuve Supposons l'existence d'un adversaire A contre cette signature avec une probabilité de succès non négligeable. Pour fonctionner, cet adversaire interroge un oracle aléatoire.

Construisons un adversaire B contre le caractère à sens unique de la fonction RSA_e. Cet adversaire reçoit de l'oracle O un défi $y \in \mathbb{Z}/n\mathbb{Z}$ qu'il doit inverser.

Pour trouver l'inverse de y, l'algorithme B va utiliser l'adversaire A comme sous-programme et va simuler auprès de lui l'oracle aléatoire du système de signature. Lorsque l'algorithme B reçoit de A un message m_i, il choisit une valeur $H(m_i) = y_i$ aléatoire, mais différente de y. Il soumet y_i à l'oracle O et reçoit en échange la valeur inverse $\sigma_i = \mathrm{RSA}_e^{-1}$ qu'il fournit en réponse à A comme valeur de la signature de m_i.

Lorsque finalement A doit proposer son message m et sa signature forgée, l'algorithme B fournit la valeur $H(m) = y$ comme valeur de l'oracle aléatoire pour m. Soit σ la signature proposée par l'algorithme A. L'algorithme B propose cette valeur comme inverse de $\mathrm{RSA}_e^{-1}(y)$.

Par construction le succès de A équivaut au succès de B. Les deux algorithmes ont donc la même probabilité de succès. La probabilité de succès de B est donc non négligeable, ce qui contredit l'hypothèse selon laquelle la fonction RSA_e est à sens unique. L'adversaire A n'existe donc pas, ce qui prouve que ce système de signature est sûr contre une attaque à messages choisis. □

Le modèle de l'oracle aléatoire rend stérile toute attaque à messages choisis. L'interrogation de l'oracle pour établir la signature des messages choisis n'apporte aucune information à l'adversaire. Lorsqu'il devra forger sa propre signature, la valeur de l'oracle sera aléatoire et indépendante de toutes les valeurs qu'il aura sollicitées auparavant pour les messages choisis.

Exercice 7.2. *La signature ElGamal* Ce système de signature été proposé par le chercheur Taher ElGamal en 1985, reposant sur la difficulté du problème du logarithme discret dans le groupe multiplicatif de $\mathbb{Z}/p\mathbb{Z}$.

Soit p un nombre premier et g un générateur du groupe multiplicatif $(\mathbb{Z}/p\mathbb{Z}^\star, \times)$. Le système de signature ElGamal tel qu'il a été publié dans l'article original est défini ainsi :

1. La clé privée est un élément aléatoire k_{priv} de $\mathbb{Z}/(p-1)\mathbb{Z}$.
2. La clé publique est $k_{\mathrm{pub}} = g^{k_{\mathrm{priv}}}$.
3. La signature d'un message $m \in \mathbb{Z}/(p-1)\mathbb{Z}$ est calculée ainsi :
   (a) Choisir un élément r aléatoire dans $\mathbb{Z}/(p-1)\mathbb{Z}$ et premier avec $p-1$, puis calculer $u = g^r$.
   (b) Résoudre l'équation linéaire $m = k_{\mathrm{priv}}u + rs$, d'inconnue s dans $\mathbb{Z}/(p-1)\mathbb{Z}$.
   (c) La signature du message m est le couple $\sigma = (u, s) \in \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/(p-1)\mathbb{Z}$.
4. La vérification de la signature $\sigma = (u, s)$ d'un message m consiste à vérifier l'égalité suivante dans $\mathbb{Z}/p\mathbb{Z}$.

$$g^m = k_{\mathrm{pub}}^u \times u^s$$

a) Vérifier que la signature d'un message correctement signé passe toujours la fonction de vérification.
b) Montrer que la signature ElGamal n'est pas sûre, y compris lorsque l'adversaire de dispose d'aucun message signé.
Indication : Soient e et v des éléments quelconques de $\mathbb{Z}/(p-1)\mathbb{Z}$. Poser $u = g^e k_{\mathrm{pub}}^v$ et $s = -u/v$, puis trouver un message m tel que (u, s) est une signature valide de m.

Pour rendre inefficace l'attaque présentée dans cet exercice, il convient d'utiliser le paradigme *hache-puis-signe* en remplaçant, dans le calcul et la vérification, le message m par son image par un oracle aléatoire h : $(u, m) \mapsto h(u, m)$, mais la sécurité de ce schéma n'est pas prouvée.

Exercice 7.3. *Signer un symbole binaire avec une couple de bijections sans pince à trappe*

Une pince (*claw*) d'un couple de bijections f_0 et f_1 sur un même ensemble E est un couple (x, y) d'éléments de E tel que $f_0(x) = f_1(y)$. On dit qu'une famille de bijections est sans pince (*claw free*) s'il est difficile de trouver une pince pour tout couple d'entre elles.

a) Soit m un module RSA et e un exposant public. Montrer que si la fonction RSA_e est à sens unique, alors il est possible de construire un couple (f_0, f_1) de bijections sans pince avec trappe, c'est-à-dire qu'il est possible de construire une pince connaissant la trappe.

b) Soit (f_0, f_1) un couple de bijections sur E sans pince à trappe. On considère le système de signature défini sur l'espace des messages binaires $\mathcal{M} = \{0, 1\}$:

- La clé privée est une pince (x, y) pour le couple (f_0, f_1).
- La clé publique est le couple (f_0, f_1) et l'élément z de E tel que $f_0(x) = f_1(y) = z$.
- La signature du message $b \in \{0, 1\}$ est x si $b = 0$ et y si $b = 1$.

Comment vérifier la signature ?

c) Démontrer que s'il est difficile de trouver une pince sans la trappe, alors il est difficile de forger une signature pour un message $b \in \{0, 1\}$ lors d'une attaque à message choisi.

2 Signature et preuve sans divulgation de connaissance

L'authentification fait intervenir deux acteurs : un prouveur qui doit s'identifier et prouver son identité avant d'accéder à un service et un vérificateur chargé de contrôler que le prouveur dispose bien du droit d'y accéder. Un des moyens mis en œuvre dans le monde numérique est de demander à la personne de prouver qu'elle dispose bien d'une donnée secrète qui la caractérise. Plusieurs méthodes permettent d'arriver à cet objectif :

Authentification statique On demande à la personne de révéler son secret. C'est le moyen le plus simple et il est utilisé par exemple dans la téléphonie mobile pour identifier le porteur de la carte SIM qui doit introduire son code d'identification personnel (PIN *Personnal Identification Number*) avant d'accéder au service. Le même mécanisme est utilise pour le payement par carte bancaire. Malheureusement, dès que le secret est révélé, ce n'en est plus un et nous faisons tous confiance au système pour qu'il ne divulgue pas l'information.

Authentification dynamique Ce mécanisme d'authentification interactif fonctionne sur le mode *défi - réponse*. Le vérificateur lance un défi auquel le prouveur doit répondre par une réponse qui dépend directement du secret, comme par exemple le chiffrement du défi à l'aide d'une clé. L'inconvénient de ce mécanisme est qu'à force d'être utilisé, le secret s'use, et si le défi est aléatoire, il devient de moins en moins probable qu'il ne soit pas déjà utilisé, et un observateur attentif peut enregistrer toutes les authentification et espérer pouvoir rejouer l'une d'entre elle.

Authentification sans divulgation Les systèmes de preuves interactives sans divulgation de connaissances sont été développés à partir de 1985 pour pallier ce défaut et de pouvoir s'authentifier sans révéler la moindre information sur le secret détenu. Leur présentation est l'objet du présent paragraphe.

En 1986, les chercheurs Amos Fiat (né en 1956) et Adi Shamir (né en 1952) ont publié une technique qui permet de transformer les protocoles de ce type en un système de signature, qui est par nature non interactif. Pour cela, ils ont dû faire appel à un oracle aléatoire.

La suite de ce chapitre présente quelques exemples de systèmes de preuve sans divulgation et les signatures associées.

2.1 Système de preuve sans divulgation de connaissance

Un système de preuve sans divulgation de connaissance (*Zero Knowledge proof*) est un protocole interactif entre deux acteurs qui permet à l'un d'eux – le prouveur – de prouver qu'il dispose d'une information secrète, de telle sorte qu'à l'issue du protocole, son partenaire – le vérificateur – en soit convaincu. Ce qui permet au vérificateur d'asseoir sa conviction est un témoin de la donnée secrète sous la forme de son image par une fonction à sens unique. Après les échanges avec le prouveur, le vérificateur

a acquis la certitude que le prouveur a bien la connaissance du secret, mais aucune parcelle d'information n'en a été divulguée. Le vérificateur n'a rien appris au cours de l'échange.

Par ailleurs la preuve n'est pas transmissible. Le vérificateur, lui même convaincu que le prouveur connait l'information, ne peut transmettre sa conviction à personne d'autre.

Plus précisément, soit x une donnée secrète, dont seule est connue son image $y = f(x)$ par une fonction à sens unique f. Un système de preuve de la connaissance de x sans divulgation se réalise par un protocole en trois passes et quatre étapes :

1. **Engagement.** Le prouveur envoie au vérificateur une valeur u qui l'engage et sur laquelle il ne peut revenir.
2. **Défi.** Le vérificateur transmet au prouveur un défi aléatoire auquel le prouveur devra répondre.
3. **Réponse.** Le prouveur calcule alors la réponse z au défi, réponse qu'il est seul à pouvoir calculé avec les paramètres secrets, et qu'il transmet au vérificateur.
4. **Vérification :** Le vérificateur vérifie, à l'aide des paramètres publics, que la réponse reçue est conforme à l'engagement, au défi et au témoin y du secret du prouveur : $\text{Verif}(z, h, y, u) \in \{0, 1\}$. Les valeurs 0 et 1 témoignent respectivement du la réussite ou de l'échec de l'authentification.

Les propriétés attendues pour un tel protocole sont les suivantes :

— **La consistance :** un prouveur honnête doit avoir une très forte probabilité de s'authentifier ;
— **La robustesse :** un prouveur malhonnête qui ne connaît pas le secret doit avoir une très faible probabilité de s'authentifier ;
— **Aucune divulgation de connaissance :** aucune information sur le secret détenu par le prouveur ne doit transparaître du protocole. Les authentifications successives n'usent pas le secret du prouveur.

Définition 2.1. adversaire d'un protocole de preuve sans divulgation

Un adversaire contre un protocole de preuve sans divulgation de connaissance est un algorithme dont l'objectif est de s'authentifier sans connaître le secret x.

La performance d'un adversaire se mesure à sa probabilité de succès.

Définition 2.2.

On dit qu'un protocole de preuve est sûr si tout adversaire polynomial a une probabilité de succès négligeable.

Voici quelques exemples de protocoles de preuve interactive sans divulgation de connaissance qui utilisent les trois fonctions à sens unique EXP, RSA_e et SQR issus de la théorie des nombres et présentés page 10.

Preuve de connaissance du logarithme discret

Soit $(G, \times)$ un groupe cyclique d'ordre q premier, et soit g un générateur de G. On suppose bien sûr que le problème du logarithme discret dans ce groupe est difficile.

Le secret du prouveur est un élément x de $\mathbb{Z}/q\mathbb{Z}$ et le témoin public de ce secret est l'élément y de G égal à $y = g^x$.

Le protocole qui suit permet au prouveur de convaincre le vérificateur qu'il connaît la valeur x du logarithme de y en base g sans en divulguer la moindre information.

1. **Engagement.** le prouveur choisit un élément r de $\mathbb{Z}/q\mathbb{Z}$ au hasard et transmet au vérificateur sa puissance en base g, soit $u = g^r$.
2. **Défi.** le vérificateur choisit un élément aléatoire h dans $\mathbb{Z}/q\mathbb{Z}$, qu'il transmet au prouveur.
3. **Réponse.** le prouveur calcule ensuite la réponse $z = r + hx \bmod q$ qu'il transmet au vérificateur. Cette réponse est le produit du secret par le défi masqué par la quantité aléatoire qui a servi à élaborer l'engagement.

4. **Vérification :** le vérificateur vérifie alors que $g^z = u \times y^h$. En effet, $g^z = g^{r+hw} = g^r \times g^{hx} = u \times y^h$.

Montrons que ce protocole est sans divulgation est qu'il est sûr.

Proposition 2.3.

La preuve interactive de connaissance du logarithme en base g de y décrit ci-dessus ne divulgue aucune information sur $x = \log_g(y)$.

De plus, si le calcul du logarithme de y en base g est difficile, alors ce protocole est sûr.

Preuve Si le prouveur peut prévoir la valeur du défi h, alors il peut choisir une réponse arbitraire z et calculer l'engagement $u = g^z/y^h$ qui correspond à cette réponse. Ceci montre qu'un échange peut être simulé sans connaître le secret x. Rien ne distingue l'échange simulé d'un véritable échange. D'une part l'échange de révèle rien sur x, mais l'observation de l'échange de prouve aucunement la détention du secret. la conviction du vérificateur repose sur l'ordre des opérations.

Montrons maintenant la sécurité. Supposons l'existence d'un adversaire A contre ce protocole qui peut s'authentifier sans connaître la valeur du secret x avec une probabilité de succès non négligeable. Montrons qu'on peut utiliser cet adversaire pour élaborer un algorithme de calcul du logarithme en base g de y. L'adversaire A est un algorithme au comportement probabiliste dont la première étape est de tirer une valeur r aléatoire. Rappelons qu'un tel algorithme peut être considéré comme un algorithme déterministe auquel on passe comme paramètre une chaîne aléatoire s qui définira son comportement probabiliste. Considérons l'algorithme B suivant qui utilise A deux fois comme sous-programme :

Algorithme 2.4. calcul du logarithme discret de y

1. Appeler l'algorithme A avec une chaîne aléatoire s.
   - Soit u la valeur d'engagement transmise par A.
   - B transmet à A un défi aléatoire $h \in \mathbb{Z}/q\mathbb{Z}$.
   - Soit z la réponse transmise par A.
2. Appeler une seconde fois l'algorithme A avec la même chaîne aléatoire s.
   - La valeur d'engagement u transmise par A sera par conséquent la même qu'auparavant.
   - B transmet à A un défi aléatoire $h' \in \mathbb{Z}/q\mathbb{Z}$, indépendant du défi précédent.
   - Soit z' la réponse transmise par A.
3. Si les deux réponses de A passent la vérification, et si les deux défis h et h' sont différents, alors renvoyer $x^\star = (z - z')/(h - h') \bmod q$.

   Sinon, renvoyer une valeur $x^\star$ aléatoire dans $\mathbb{Z}/q\mathbb{Z}$.

Soit p la probabilité de succès de A. Comme les valeurs des deux défis sont aléatoires et indépendantes, la probabilité de succès de A aux deux appels est p^2 qui reste non négligeable. Dans le cas où les deux réponses z et z' de A passent la vérification, elles satisfont :

$$\begin{cases} g^z &= g^{hx} \times u \\ g^{z'} &= g^{h'x} \times u \end{cases}$$

Par division membre à membre et passage au logarithme, on a $z - z' = x(h - h')$. Dans ce cas, si $h \neq h'$, la valeur rendue par l'algorithme B vaut bien le logarithme de y en base g. Le succès de A lors des deux appels implique donc le succès de B, sauf en cas d'égalité de h et h', ce qui survient avec une probabilité négligeable égale à $1/q$. Il en résulte une probabilité de succès de B est supérieure à $p^2 - 1/q$, ce qui est non négligeable, et contredit l'hypothèse de difficulté de calcul de $\log_g(y)$. □

Exercice 7.4. Pourquoi ne faut-il pas utiliser deux fois la même valeur aléatoire r pour s'authentifier par cette preuve interactive ?

Exercice 7.5. *Preuve de connaissance d'un antécédent pour la fonction RSA*

Soit n un module RSA, soit e un exposant public et d l'exposant privé correspondant. On suppose que l'exposant public e a environ la même taille que le module n. Le secret du prouveur est un élément x de $\mathbb{Z}/n\mathbb{Z}$ et le témoin est l'image $y = x^e \bmod n$ de x par la fonction RSA_e. On considère la preuve de connaissance de x suivante :

1. Engagement. Le prouveur choisit au hasard un élément r de $\mathbb{Z}/n\mathbb{Z}$ et transmet au vérificateur $u = r^e$.
2. Défi. Le vérificateur choisit un élément aléatoire h dans l'ensemble $\{1, \ldots, e-1\}$ qu'il transmet au prouveur.
3. Réponse. Le prouveur calcule alors la réponse $z = r \times x^h$ qu'il renvoie au vérificateur.
4. Vérification : Le vérificateur vérifie alors que $z^e = u \times y^h$.

a) Montrer que cet échange ne divulgue aucune information sur x.
b) Montrer comment retrouver x à partir de la connaissance des réponses à deux défis distincts h et h' pour le même engagement u. En déduire une preuve de la sécurité de ce protocole.

Exercice 7.6. *Preuve de connaissance d'une racine carrée modulo un entier composite*

Soit n un entier naturel égal au produit de deux nombres premiers distincts p et q. Le secret du prouveur est un élément $x \in \mathbb{Z}/n\mathbb{Z}$, le témoin est son carré $y = x^2$. On considère la preuve de connaissance de x suivante :

1. Engagement. Le prouveur choisit un élément r de $\mathbb{Z}/n\mathbb{Z}$ au hasard et transmet au vérificateur son carré $u = r^2$.
2. Défi. Le vérificateur choisit un élément aléatoire h dans l'ensemble $\{1, \ldots, n-1\}$ qu'il transmet au prouveur.
3. Réponse. Le prouveur calcule alors la réponse $z = r \times x^h$ qu'il renvoie au vérificateur.
4. Vérification : Le vérificateur vérifie alors que $z^2 = u \times y^h$.

a) Montrer que cet échange ne divulgue pas d'information sur x.
b) Donner des arguments de sécurité pour ce protocole.

2.2 Le paradigme de Fiat-Shamir

Le paradigme de Fiat-Shamir (1986) est une méthode générale pour construire un système de signature asymétrique à partir d'un protocole d'authentification à divulgation nulle de connaissance et d'un oracle aléatoire.

Considérons une preuve interactive de la connaissance d'une donnée privée x, déposée par l'image y de x par une fonction à sens unique f selon le protocole général exposé au paragraphe précédent. Ce protocole peut définir un système de signature à clé publique de la façon suivante :

- La clé privée de signature est le secret x détenu par le prouveur et la clé publique correspondante est son image par f : $k_{\text{priv}} = x$ et $k_{\text{pub}} = f(x) = y$.
- La fonction de production de signature consiste à engager une valeur u et à répondre au défi égal à l'image du message à signer m et de la valeur u engagée par un oracle aléatoire : $h = H(m, u)$. La signature est alors le couple $\sigma = (u, z)$, où u est la valeur engagée et z est la réponse du prouveur.
- La vérification de la signature consiste à appliquer la fonction de vérification du système de preuve : vérifier que $\mathrm{Verif}(z, h, y, u) = 1$.

Le défi h dépend du message à signer, mais aussi de l'engagement u, afin que le prouveur ne puisse revenir dessus. Dans le cas contraire, le défi est prévisible puisqu'il ne dépend que du message à signer, et il est possible de construire un défi et une réponse, et donc de forger une signature, sans connaître la clé privée.

Montrons la sécurité du système de signature ainsi défini.

Proposition 2.5. sécurité du paradigme de Fiat-Shamir

Si le protocole de preuve interactive est sûr, alors il en est de même pour le système de signature construit à partir de celui-ci.

Preuve Démontrons la contraposée de cette implication et supposons l'existence d'un adversaire A contre le système de signature dont la probabilité de succès est non négligeable. Construisons un prouveur P qui passera la vérification auprès d'un vérificateur V avec une probabilité de succès non négligeable.

Le modèle de l'oracle aléatoire rendant vaine l'attaque à messages choisis, on peut supposer que l'adversaire fournit juste un message m avec une signature forgée $\sigma = (z, u)$.

Le prouveur tient le rôle de l'oracle aléatoire. Pour construire sa signature, A a besoin de la valeur $H(m, u)$. Il transmet à P le couple (m, u). Le prouveur P transmet alors u au vérificateur comme engagement.

Le prouveur P reçoit du vérificateur un défi h qu'il renvoie à A comme valeur $h = H(m, u)$. Ensuite, A fournit la signature forgée $\sigma = (u, z)$ du message m. Le prouveur renvoie alors au vérificateur la réponse z.

Par construction, le succès de A implique le succès de P, ce qui montre que $P_{\text{succes}}(A) \leq P_{\text{succes}}(P)$. Si le premier membre n'est pas négligeable, alors le second membre ne l'est pas non plus, ce qui montre l'existence d'un adversaire au système de preuve avec une probabilité de succès non négligeable. Le système de preuve n'est donc pas sûr, et c'est ce qu'il fallait démontrer. □

Voyons comment ce paradigme permet de définir un système de signature à partir du système de preuve de connaissance du logarithme discret présenté plus haut. Ce système de signature est dû au mathématicien et cryptologue Clauss Schnorr (né en 1943).

Soit G un groupe cyclique d'ordre q premier et soit g un générateur.

- La clé privée k_{priv} est un élément aléatoire de $\mathbb{Z}/q\mathbb{Z}$.
- La clé publique k_{pub} est sa puissance en base g, soit $k_{\text{pub}} = g^{k_{\text{priv}}}$.
- Pour signer un message m :
  - Choisir un élément aléatoire r dans $\mathbb{Z}/q\mathbb{Z}$ et calculer $u = g^r$.
  - Calculer $z = k_{\text{priv}} - H(m, u) \times x$, où H est un oracle aléatoire $\mathcal{M} \times G \to \mathbb{Z}/q\mathbb{Z}$, et $\mathcal{M}$ est l'espace des messages.
  - La signature est le couple $\sigma = (u, z)$.
- Pour vérifier la signature $\sigma = (u, z)$ du message m, calculer $g^z \times {k_{\text{pub}}}^{H(m,u)} = u^\star$ et vérifier que $u^\star = u$.

La sécurité de ce système de signature est une conséquence directe de la proposition 5 ci-dessus. Mais une preuve directe se calque exactement sur la preuve de sécurité du système de preuve sur lequel il s'appuie.

Le NIST (*National Institute of Standard and Technology*) américain a normalisé en 1993 une variante assez proche appelée DSA (*Digital Signature Algorithm*). Une des différence est d'avoir remplacé le hachage du message m et de l'engagement u par le hachage du message m seul. On ne connaît pas d'attaque contre le DSA, mais cette modification mineure a pour conséquence de rendre invalide la preuve de sécurité.

Exercice 7.7. Définir un système de signature dérivé de la preuve de connaissance d'un antécédent de la fonction RSA décrit dans l'exercice 5.

Exercice 7.8. Définir un système de signature dérivé de la connaissance d'une racine carrée modulo un entier composite décrit dans l'exercice 6.

3 Les cinq mondes d'Impagliazzo

Le premier mécanisme de signature publié est le RSA, en 1978, dans l'article *A method for Obtaining Digital Signatures ans Public-key Cryptosystems* par Ronald Rivest (né en 1947) , Adi Shamir (né en 1952) et Leonard Adleman (né en 1945) [10]. Cette publication a fait suite à la description du RSA comme mécanisme de chiffrement dans l'article *A new kind of cipher that would take millions of years to break*, paru en août 1977 dans la rubrique des jeux mathématiques de Martin Gardner de la revue *Scientific American*.

La signature RSA a été présentée comme un « *chiffrement avec la clé privée* », en inversant le rôle de la clé publique et de la clé privée. La signature est vérifiée en la « *déchiffrant avec la clé publique* ». Cette description montre une symétrie entre chiffrement à clé publique et signature. Pendant plusieurs années qui ont suivi ces publications, la signature a été pensée comme duale du chiffrement à clé publique, ancrant l'idée que la fonction de signature nécessite une fonction à sens unique avec trappe, tout comme en a besoin le chiffrement à clé publique. **Cette approche est fausse !**

D'une part, il n'est pas toujours possible de transformer un système de confidentialité à clé publique en mécanisme de signature. Par exemple si le chiffrement n'est pas déterministe, comme c'est le cas du système de McEliece, les cryptogrammes ont une forme particulière et il n'est pas toujours possible d'invoquer la clé privée sur un message arbitraire. D'autre part, nous montrerons dans ce chapitre qu'il est possible de construire un système de signature numérique à partir d'une simple fonction à sens unique, sans nécessité d'une trappe. La signature à clé publique appartient au même monde que le chiffrement symétrique.

Le chiffrement à clé publique, lui, nécessite une primitive plus évoluée : celle de fonction à sens unique avec trappe, et qui sera traité dans le chapitre suivant. Ainsi, la frontière n'est pas entre cryptographie symétrique et asymétrique, mais entre cryptographie qui nécessite ou non une trappe.

La difficulté attendue des problèmes cryptographiques est une difficulté *en moyenne*, théorie développée par Leonid Levin (né en 1948) qui a inspiré au chercheur Russel Impagliazzo (né en 1963) [7] la possibilité de cinq mondes qui diffèrent selon le type de problème qu'il y est possible de résoudre efficacement. Ces mondes sont issus de notre ignorance actuelle de la réponse à des grandes conjectures de la théorie de la complexité.

Impagliazzo a vulgarisé ces conjectures par une visite guidée de cinq mondes possibles selon les réponses qui peuvent être apportées à ces conjectures. Ces mondes sont illustrés par la cryptographie qu'il y est possible de réaliser. Il met aussi en scène le personnage de Grouse [1], professeur imaginaire du jeune élève Gauss, mais inspiré d'un fait réel. Grouse, voulant exercer ses élèves à l'usage de l'addition, leur a demandé de calculer la somme des entiers de 1 à 100. Il espérait avec cette colle avoir quelques moments tranquilles, mais c'était sans compter sur la présence du brillant élève dans sa classe qui a trouvé rapidement la réponse par un raisonnement astucieux. Le pauvre professeur s'en est trouvé humilié. Impagliazzo raconte que ce dernier a passé le reste de sa vie à chercher en vain un problème qui pourrait mettre fin à l'arrogante réussite de son élève, pour finir misérablement sa vie dans une asile d'aliénés.

3.1 Algorithmica

Ce monde est celui dans lequel $\mathcal{P} = \mathcal{NP}$. Dès l'instant où il existe un algorithme efficace pour vérifier la solution à un problème donné, il existe également une méthode efficace pour en produire la solution. Grouse ne peut pas coller Gauss avec un problème dont il pourrait présenter la solution à toute la classe, puisque ce dernier peut trouver la solution directement à partir de la vérification. L'usage de l'informatique s'en trouve révolutionné. L'ordinateur peut se charger de tâches usuellement dévolues aux seuls humains. Parler une langue naturelle se réduit à savoir lire et interpréter un corpus réduit de textes. Savoir multiplier permet de factoriser efficacement. Déchiffrer et décrypter, avec ou sans clé, sont des problèmes de difficultés équivalentes. Aucune cryptographie n'y est possible autre que celle du masque jetable de Vernami [2]. Dans ce monde, il est impossible de réserver l'accès à une information à certains sans que tous ne puissent également y accéder. Le prix à payer à savoir résoudre tous les problèmes algorithmiques de la vie courante est l'impossibilité d'y conduire la moindre cryptographie efficace.

1. *to grouse* : rouspéter
2. Ce mécanisme de chiffrement des téléscripteurs consiste à combiner chaque caractère du message en clair avec un caractère aléatoire présent sur une bande de papier partagée entre l'émetteur et le destinataire. Il a été utilisé en particulier dans le téléphone rouge, mis en place à partir d'octobre 1962 pour détendre les relations entre les deux grandes puissances après la crise des missiles de Cuba. Ce mécanisme est doté d'une sécurité inconditionnelle, mais son usage est impraticable dans un réseau ouvert comme Internet, car il nécessite l'échange préalable à toute communication d'autant de quantité d'aléa

3.2 Heuristica

Heuristica est un monde où les problèmes $\mathcal{NP}$ ne sont difficiles à résoudre que dans le pire des cas. En moyenne, la résolution reste accessible à un algorithme efficace. En pratique, ce monde est presque en tout points comparable à Algorithmica. La différence réside dans l'existence de rares problèmes pratiquement insolubles. Supposons par exemple que la recherche d'un problème mathématique prenne un temps t, et sa résolution un temps $2t$. Comme tout chercheur le sait, la réponse à un problème conduit inévitablement à un nouveau problème. En Heuristica, ce deuxième problème, qui survient au bout d'un temps $3t$, prend un temps $6t$ à être résolu. La récurrence est géométrique, et en quelques itérations la frontière de l'infaisable est rapidement atteinte. Les instance difficiles des problèmes $\mathcal{NP}$ existent, mais sont aussi difficiles à trouver. Le temps moyen pour résoudre un problème $\mathcal{NP}$ reste comparable à celui pour en concevoir l'énoncé. Grouse peut coller Gauss avec un problème difficile, mais il lui faudra au moins deux fois plus de temps pour l'établir qu'à Gauss pour le résoudre – n'oublions pas de Gauss est un élève particulièrement brillant. D'un point de vue cryptographique, le temps passé à trouver un chiffre correspond au temps pendant lequel le secret est garanti. En Heuristica, il n'y a pas plus d'espoir qu'en Algorithmica de concevoir une cryptographie efficace.

3.3 Pessiland

Selon Impagliazzo, ce monde est le pire qui soit. Il existe des problèmes difficiles à résoudre en moyenne, mais pour toute fonction efficacement calculable, il existe une façon efficace de trouver un antécédent à une valeur donnée. En d'autres termes, il n'existe pas en Pessiland de fonction à sens unique. Il est facile de produire des instance difficiles de problèmes $\mathcal{NP}$, mais il n'est pas possible de produire des instances difficiles de problèmes dont la solution soit connue. En Pessiland, Grouse peut poser à Gauss une colle insoluble, mais personne, y compris la grande clairvoyance de Gauss ne pourra conduire à une solution. L'humiliation de Grouse restera entière lorsque la classe demandera la réponse au professeur qui aura toutes les difficultés du monde pour en exhiber une. Rappelons que la théorie cryptographique fonde le chiffrement symétrique sur un axiome d'existence de fonctions à sens unique. Cette cryptographie ne fait pas partie de Pessiland. De nombreux problèmes de la vie courante y restent difficiles, mais aucun d'entre eux ne peut être utilisé à des fins cryptographiques. Ce monde agaçant cumule tous les désavantages.

3.4 Minicrypt

En Minicrypt, les fonctions à sens unique existent, mais n'ont pas de trappe. La cryptographie symétrique est possible, mais pas la cryptographie à clé publique. Deux correspondants devront préalablement s'accorder sur une clé secrète, fusse-t-elle de taille réduite, pour pouvoir échanger par la suite de façon tout à fait discrète une très grande quantité d'informations sur un canal public. Une fonction à sens unique f peut être utilisée pour produire un problème difficile dont on connaît une solution, par exemple en tirant un élément x au hasard et en calculant $y = f(x)$ et en posant le défi de trouver un antécédent à y. Dans ce monde, Grouse garde la tête haute, car il peut poser à Gauss un problème que ce dernier aura bien du mal à résoudre. Il pourra même exhiber fièrement la solution face à la classe ébahie et remporter une certaine victoire. La théorie cryptographique montre qu'en Minicrypt, il est aussi possible de produire de signatures à clés asymétriques : une clé privée pour produire la signature et une clé publique pour sa vérification.

3.5 Cryptomania

Cryptomania est le monde le plus proche de celui dans lequel nous vivons aujourd'hui. Il y existe des fonctions à sens unique avec trappe. Il s'agit de fonctions dont les valeurs sont efficacement calculables par tous, les antécédents sont difficiles à exhiber, sauf pour les détenteurs d'une information supplémentaire maintenue secrète. La cryptographie à clé publique y est possible. Ainsi, tout le monde peut chiffrer un message, mais le déchiffrement reste réservé au seul détenteur de la trappe qui tient lieu de clé privée. Deux correspondants peuvent s'accorder sur un secret partagé en commun à partir de données qu'ils échangent publiquement. En Cryptomania, Grouse peut enfin asseoir son autorité de professeur

en posant à la classe entière un problème pratiquement impossible à résoudre. Mieux ! Il peut humilier Gauss en révélant au reste de la classe une indication qui permettra aux autres élèves d'accéder à la solution, alors que pour le pauvre Gauss, resté dans l'ignorance de cette indication, le problème restera définitivement insoluble. En Cryptomania, les possibilités de la cryptologie n'ont de limites que celles de l'imagination des concepteurs : vote électronique, monnaie digitale anonyme, manipulation de données chiffrées. Le niveau d'intimité de la sphère privée n'est pas limité par la technique, mais seulement par les décisions sociales ou politique qui dictent la détention des clés privées.

Même s'il est théoriquement possible de construire une fonction de signature à clés asymétriques en Minicrypt, la plupart de systèmes de signature actuels, comme la signature RSA, l'algorithme DSA *Digital Signature Algorithm*, ElGamal, Schnorr, ECDSA, *etc.* sont construits à partir de fonctions à sens unique particulières issues de la théorie des nombres avec des hypothèses qui les font appartenir au monde Cryptomania.

Un des principaux objet de la suite de ce chapitre est de construire une signature numérique dans le monde Minicrypt.

4 La signature à clé jetable jetable de Lamport

La signature définie dans ce paragraphe est une signature à clé jetable, utilisable une fois seulement pour signer un seul document. Elle est construite à partir d'une fonction à sens unique sur l'ensemble $\{0,1\}^n$, pour la valeur n du paramètre de sécurité. Ce schéma de signature appartient donc à Minicrypt. Mais la clé étant jetable, pour signer un autre document, il faudra utiliser un autre couple de clés. L'idée de cette signature figure dans l'article de Diffie et Hellman de 1976, mais a été publié en 1979 par l'informaticien Leslie Lamport (né en 1941). Voir [9].

La signature à clé jetable de Lamport signe des messages qui sont des mots de ℓ symboles binaires, l'entier ℓ étant majoré par un polynôme en n. On suppose que l'on dispose d'une fonction à sens unique $f : \{0,1\}^n \to E$.

Définition 4.1. Schéma de signature à clé jetable de Lamport

Pour la valeur n du paramètre de sécurité, le système de signature de Lamport est défini par les éléments suivants :

- L'ensemble $\mathcal{M}$ des messages est l'ensemble $\{0,1\}^\ell$.
- La clé privée est constituée de 2ℓ éléments de $\{0,1\}^n$ aléatoires :

$$k_{\text{priv}} = \begin{pmatrix} x_{01} & x_{02} & \cdots & x_{0\ell} \\ x_{11} & x_{12} & \cdots & x_{1\ell} \end{pmatrix}$$

- La clé publique est constituée des images des composantes de la clé privée par la fonction à sens unique f :

$$k_{\text{pub}} = \begin{pmatrix} y_{01} & y_{02} & \cdots & y_{0\ell} \\ x_{11} & y_{12} & \cdots & x_{1\ell} \end{pmatrix}$$

  où pour $i \in \{0,1\}$ et $j \in \{1,\ldots,k\}$, on a $y_{ij} = f(x_{ij}) \in E$.
- La signature du message $m = m_1 \cdots m_\ell$ est $\text{Sign}(m) = (x_{m_1 1}, \ldots, x_{m_\ell \ell})$.
- La vérification de la signature $\sigma = (\sigma_1, \ldots, \sigma_\ell)$, consiste à vérifier que pour tout indice $i \in \{1,\ldots,\ell\}$, on a $f(\sigma_i) = y_{m_i i}$.

Signer un message révèle certains antécédents x_{bi} de la clé publique par la fonction à sens unique. Pour l'indice i, si la composante m_i vaut 0, c'est la valeur x_{0i} qui est révélée, et si m_i vaut 1, c'est la composante x_{1i}.

Vérifier la signature consiste à vérifier que les valeurs des composantes de la signatures valent bien celles correspondantes dans la clé publique. La signature est considérée comme valide si toutes compo-

santes la signature sont ainsi correctement vérifiées selon les valeurs binaires des composantes du message.

Signer un message révèle une partie de la clé privée. Avec seulement les signatures de deux messages complémentaires l'un de l'autre, toute la clé privée est révélée. La clé privée n'est donc utilisable que pour signer un seul message.

L'intérêt de ce système de signature est surtout théorique. Outre que cet algorithme ne permet de signer qu'un seul document, l'inconvénient rédhibitoire est la taille des clés et des signatures. Pour un message de ℓ symboles binaires, la signature a une taille de $\ell \times n$ symboles binaires, la clé privée a une taille de $2\ell \times n$ symboles binaires. Ces tailles limitent l'intérêt pratique d'un tel mécanisme. Pour signer un message de 200 symboles binaires avec un paramètre de sécurité égal à 200 aussi, les clés publiques et privées auront une taille de 10 kilo-octets et la signature une taille de 5 kilo-octets.

Pour signer un message, l'adversaire doit pouvoir inverser la fonction f sur les valeurs y_{ij} de la clé publique. La fonction f étant supposée à sens unique, trouver les composantes de la signature est difficile. Le théorème suivant énonce cette idée de manière plus formelle.

Théorème 4.2. Sécurité de la signature à clé jetable de Lamport

Si la fonction f est à sens unique, alors le système de signature à clé jetable de Lamport est sûr.

Preuve Supposons l'existence d'un adversaire A qui, à partir de la connaissance de la clé publique, sait produire un message et sa signature. On suppose également que cet adversaire a une probabilité de succès non négligeable.

Utilisons cet adversaire pour construire un algorithme B qui trouve un antécédent à une valeur y avec une probabilité de succès non négligeable. Cet algorithme va fournir à A la clé publique en plaçant la valeur y à inverser à une position aléatoire dans l'espoir que la signature forgée par A fournisse l'antécédent cherché.

Algorithme 4.3.

Entrée : une valeur $y \in \{0,1\}^n$ dont il faut trouver un antécédent.

1. Choisir une position aléatoire $j^\star \in_R \{1,\ldots,k\}$ et une valeur binaire $i^\star \in_R \{0,1\}$.
2. B génère une clé publique en imposant la valeur y en position $(i^\star, j^\star)$. Les autres valeurs sont $y_{ij} = f(x_{ij})$ où la valeur x_{ij} est aléatoire. Cette clé publique est transmise à A.
3. L'algorithme A fournit un message m et sa signature σ.

Sortie : Si $m_{j^\star} = i^\star$ alors renvoyer $\sigma_{j^\star}$, sinon, renvoyer une valeur aléatoire.

Comme $i^\star$ et $j^\star$ sont aléatoires, l'événement $m_{j^\star} = i^\star$ survient avec une probabilité $1/2$, indépendamment du succès de A. Par ailleurs, le succès de A implique celui de B. Donc :

$$\Pr_{\text{succes}}(B) \geq \frac{1}{2} \Pr_{\text{succes}}(A).$$

La probabilité de succès de A étant supposée non négligeable, celle de B ne l'est pas non plus. □

Exercice 7.9. Démontrer que, si on autorise la signature de deux messages distincts avec la même clé, alors le système de signature de Lamport reste sûr contre une attaque à (un seul) message choisi, mais qu'il ne l'est pas pour signer trois messages.

Le schéma de signature de Lamport peut signer des messages de taille quelconque, pourvu que les clés publiques et privées aient le nombre suffisant de composantes. Il faut au moins deux termes par symbole binaire du message. Éventuellement, ces clés peuvent être produite par un générateur de pseudo aléa.

5 Signature à clé recyclable

L'inconvénient rédhibitoire de la signature à clé jetable est qu'elle ne permet de signer qu'un seul message. Pour signer un autre message, il faut une autre clé. Ce paragraphe décrit une méthode qui permet de pallier ce défaut. La méthode décrite est applicable à tout système de signature asymétrique à clé jetable comme l'est celui décrit au paragraphe précédent.

Pour cela, on utilise une arborescence de certificats, c'est-à-dire une clé publique et sa signature, à l'instar des infrastructures de clés publiques. Ainsi, la signature d'un message binaire $m = m_1 \cdots m_\ell$ consiste à parcourir l'arbre binaire à partir de la racine selon la valeur de chaque terme du message. Lors de la lecture d'un 0, l'arbre est parcouru à gauche et lors de la lecture d'un 1, il est parcouru à droite. À chaque nœud de l'arbre, deux couples de clés est généré : un pour le fils gauche et un autre pour le fils droit. Les clés publiques sont signées avec la clé privée du niveau précédent. Les clés de la racine sont signées avec la clé initiale du signataire. Ces clés et leur signature constitue un certificat. La signature du message est constituée de la suite de chacun des certificats. La clé initiale du signataire ne sert donc qu'à signer le premier certificat de la signature qui est toujours identique pour tous les messages. Selon la valeur du premier symbole du message, c'est l'une ou l'autre des deux clés produites qui servira à signer le certificat suivant.

Les couples de clé sont générés à partir d'un générateur de pseudo aléa dont le germe initial fait partie de la clé privée du signataire. À chaque nœud de l'arbre ce générateur produit deux germes, un pour chacun des deux fils. Ainsi, si deux messages ont le même préfixe, les clés produites seront identiques, produisant en conséquence des certificats identiques dans la signature.

Pour vérifier la signature du message, le certificat racine est vérifié avec la clé initiale. Les clés publiques qu'il contient servent à vérifier le certificat du niveau suivant. La signature du message est considérée comme valide si tous les certificats sont correctement vérifiés.

Le dernier certificat qui correspond au dernier symbole binaire du message ne nécessite pas de clé publique, puisqu'il n'y a pas de niveau suivant. Il peut ne contenir que la signature d'un message réduit à un seul symbole binaire, signé avec la clé du niveau précédent.

Un principe similaire est utilisé dans les infrastructures de clés publiques pour certifier les clés publiques des utilisateurs d'un réseau afin d'en assurer l'authenticité. Les clés publiques sont signées avec la clé privée d'une autorité de certification. Au besoin, la clé publique de cette autorité est elle-même signée par une autorité de certification de rang supérieur. On se trouve alors en présence d'une arborescence de certificats dont la validité ne repose que sur l'authenticité du certificat racine. Seule la clé publique de l'autorité qui signe le certificat racine est nécessaire.

Dans le système à clé recyclable présenté ici, la validité de l'ensemble des certificats repose sur l'unique clé publique du signataire qui ne sert qu'à vérifier le premier certificat qui est toujours le même pour tous les messages. Les autres certificats dépendent de la succession des symboles binaires du message.

On suppose que l'on dispose des primitives suivantes :

- Un système de signature asymétrique à clé jetable Σ_0 constitué de :
  - Une fonction $\mathrm{Gen}_0(s)$ de génération de clés $k = (k_{\mathrm{priv}}, k_{\mathrm{pub}})$ à partir d'une graine $s \in \{0,1\}^n$.
  - Une fonction Sign_0 de production de la signature d'un message.
  - Une fonction Verif_0 de vérification des signatures.
- Un générateur de pseudo-aléa d'expansion double $g : \{0,1\}^n \to \{0,1\}^{2n}$.

Le système de signature à clé recyclable repose sur la génération pseudo-aléatoire de paires de clés pour chaque symbole binaire du message.

Ce système de signature requiert un mécanisme de signature à clé jetable ainsi qu'un générateur de pseudo aléa. Ces primitives se construisent à partir d'une fonction à sens unique et appartiennent toutes deux à Minicrypt.

Définition 5.1. Schéma de signature asymétrique à clé recyclable

- L'ensemble des messages est $\mathcal{M} = \{0,1\}^\ell$.
- La clé publique est une clé publique k^0_{pub} du système Σ_O.
- La clé privée est le couple (k^0_{priv}, s_0) où k^0_{priv} est la clé privée correspondant à k^0_{pub} et s_0 est un germe du générateur de pseudo-aléa g.
- Pour signer le message $m = m_1 \cdots m_\ell$, pour $i = 1$ jusqu'à ℓ, faire :
  1. Produire les germes pour les clés de l'itération suivantes : $g(s_{i-1}) = (x_0, x_1)$.
  2. Générer deux paire de clé avec les deux germes $\text{Gen}(x_0) = (\kappa^0_{\text{priv}}, \kappa^0_{\text{pub}})$ et $\text{Gen}(x_1) = (\kappa^1_{\text{priv}}, \kappa^1_{\text{pub}})$: une pour chaque successeur de l'arborescence.
  3. Soit $\eta = m_i \in \{0,1\}$.
     - Poser $s_i = x_\eta$ le germe de l'itération suivante.
     - Signer les clés publiques avec la clé privée courante $\sigma_i = \text{Sign}(\kappa^0_{\text{pub}} \mid \kappa^1_{\text{pub}}, k^{i-1}_{\text{priv}})$
     - Soit c_i le certificat constitué des clés publiques et de leur signature $c_i = (\kappa^0_{\text{pub}} \mid \kappa^1_{\text{pub}}, \sigma_i)$.
     - Poser $k^i_{\text{priv}} = \kappa^\eta_{\text{priv}}$ la clé privée pour l'itération suivante.
     - La signature est constituée de la liste des certificats $(c_1, \ldots, c_\ell)$.
- Pour vérifier la signature $(c_1, \ldots, c_\ell)$, utiliser une clé publique par itération. Cette clé publique est initialisée à $k^1_{\text{pub}} = k_{\text{pub}}$ la clé publique de vérification. Puis, pour $i = 1$ jusqu'à ℓ,
  1. Vérifier le certificat c_i avec la clé publique courante $\text{Verif}(\kappa^0 \mid \kappa^1, \sigma_i, k^i_{\text{pub}})$. En cas d'échec de la vérification, considérer la signature comme invalide.
  2. Soit $\eta = m_i \in \{0,1\}$. Poser $k^{i+1}_{\text{pub}} = \kappa^\eta$ la clé publique de l'itération suivante.
  3. Si toutes les vérifications sont correctes, la signature est considérée comme valide. Si une des vérifications échoue, la signature est considérée comme invalide.

En raison de la génération pseudo-aléatoire des clés, les signatures sont toujours appliquées sur les mêmes clés publiques. Si deux messages ont le même préfixe, les premier certificats seront identiques.

Le mécanisme à clé jetable de Lamport décrit au paragraphe précédent n'est pas directement applicable. Un certificat est en effet constitué de deux clés publiques et de leur signature. Mais les clés publiques pour signer des messages de ℓ symboles binaires ont une taille de $2n \times \ell$ symboles binaires. Comme il y a deux clés publique par certificat, l'algorithme de signature doit être capable de signer des messages de $4n \times \ell$ symboles binaires. L'utilisation du système de Lamport conduit à une taille de clés exponentielle en la taille du message à signer. En effet :

- La signature σ_ℓ a une taille égale au paramètre de sécurité n,
- Les données du certificat $c_{\ell-1}$ ont une taille égale à $4n^2$.
- Les données du certificat $c_{\ell-2}$ ont une taille égale à $4^2 \times n^3$.
- ...
- Les données du certificat c_1 ont une taille égale à $4^{\ell-1} \times n^{\ell+1}$.

Une solution à ce problème réside dans l'emploi d'une fonction de compression pour que l'algorithme de signature opère sur un condensé des deux clés publiques de taille fixe égale à k symboles binaires. La construction de telles fonctions, appelées fonctions de hachage, est l'objet du chapitre suivant. Pour montrer que la signature appartient bien à Minicrypt, il faudra s'appliquer à montrer que les fonctions de hachage utilisées y appartiennent également.

8

Fonctions de hachage cryptographiques

Les systèmes de signature présentés dans le chapitre précédent opèrent sur des ensembles de messages qui sont des éléments d'un groupe fini ou des mots binaires de taille fixe. Mais les signatures utilisées en pratique doivent pouvoir opérer sur des données binaires de taille arbitraires : texte, image, son, code exécutable, *etc.* Pour cette raison le paradigme *hahe-puis-signe* soumet d'abord le message à signer à une fonction de hachage, afin de produire une empreinte de taille fixe sur laquelle sera plus aisément appliquée l'algorithme de signature. Ces fonctions ont une propriété de compression, car les messages traités sont généralement d'une taille plus importante que l'empreinte produite. Les systèmes de chiffrement à clé publique font également usage de fonctions de hachage cryptographique pour résister à des attaques à cryptogrammes choisis, comme cela sera montré dans le chapitre suivant.

Les fonctions de hachage trouvent leur origine en informatique pour définir des indices de tableau à partir de clés d'accès plus complexes comme des chaînes de caractères. Une collision survient lorsque deux clés d'accès différentes conduisent au même indice. Il faut alors prévoir un traitement spécifique. La propriété attendue des fonctions de hachage est de ne pas créer plus de collisions que le nombre attendu par une répartition équiprobable des indices.

Les fonctions de hachage cryptographiques doivent en outre résister à des collisions intentionnellement provoquées par des adversaires. Les deux propriétés classiques de sécurité attendues d'une fonction de hachage cryptographique pour assurer la sécurité des signatures sont la résistance aux collisions – il doit être difficile de trouver deux paramètres distincts qui ont la même image – ou seulement la résistance au second antécédent – pour un paramètre donné, il doit être difficile de trouver un second paramètre ayant la même image.

Le modèle de la fonction de hachage parfaite est l'oracle aléatoire, (voir encadré 3, page 73), dont la valeur est imprévisible, mais il est admis que ce modèle idéal ne correspond à aucune des fonctions utilisées en pratique qui, elles, sont calculées par un algorithme.

Dans ce chapitre, il sera montré que la résistance au second antécédent est suffisante pour assurer la sécurité des systèmes de signature, et une construction sera proposée, à partir d'une fonction à sens unique. Cela achèvera l'élaboration d'un système de signature asymétrique complet à partir de fonctions à sens unique, ancrant cette primitive cryptographique dans Minicrypt.

1 Résistance aux collisions – résistance au second antécédent

L'étude de la sécurité étant asymptotique, il est nécessaire de définir des familles de fonctions indexées par la valeur d'un paramètre de sécurité. Afin de ne pas alourdir les notations, ce paramètre sera le plus souvent omis. De plus, pour une valeur donnée de ce paramètre, on définit plutôt une famille de fonctions de hachage $E \to F$ indexée par un paramètre s pouvant faire office de clé.

Définition 1.1. Famille de fonctions de hachage

Une famille de fonctions $(h_s)_{s\in\mathcal{S}}$ d'un ensemble E vers un ensemble F, où $\mathrm{Card}(E) > \mathrm{Card}(F)$, est une famille de fonctions de hachage s'il existe deux algorithmes efficaces :

1. Un algorithme Gen qui produit un index aléatoire s dans l'ensemble d'indices $\mathcal{S}$.
2. Un algorithme qui, pour tout indice $s \in \mathcal{S}$ et tout $x \in E$ calcule la valeur $y = h_s(x)$.

Une fonction de hachage idéale est une fonction dont la valeur est imprévisible tant qu'on n'en a pas calculé la valeur, qui a les même caractéristiques qu'une fonction parfaitement aléatoire. Les fonctions de

hachages usuelles sont loin d'atteindre cet idéal.

Les fonctions de hachage concrètes utilisées dans le monde réel, comme SHA1 ou MD5, sont des familles réduites à un seul élément et sans paramètre de sécurité dont on ne peut analyser la sécurité de façon asymptotique.

Les deux critères de sécurité effectifs qui sont retenus pour les fonctions de hachage cryptographiques sont la résistance aux collisions et la résistance au second antécédent.

1.1 Résistance aux collisions

Une famille de fonctions de hachage $E \to F$, où $\mathrm{Card}(E) > \mathrm{Card}(F)$, est dite résistante aux collisions s'il est difficile de trouver une collision, c'est-à-dire deux élément x et x' de E qui ont la même image par un élément de la famille. Cette résistance se mesure aux performances d'un adversaire contre les collisions.

Définition 1.2. adversaire contre les collisions

Un adversaire contre les collisions est un algorithme dont l'entrée est un indice $s \in \mathcal{S}$ et dont la sortie est un couple d'éléments x et x' de E.

L'adversaire a gagné si $h_s(x) = h_s(x')$, et il a perdu dans le cas contraire.

L'efficacité d'un adversaire se mesure à sa probabilité de succès.

Définition 1.3.

Une famille $\mathcal{H} = (h_s)_{s\in\mathcal{S}}$ est dite résistante aux collisions si la probabilité de succès de tout adversaire polynomial contre les collisions est une fonction négligeable du paramètre de sécurité.

Comme le cardinal de E est supposé être strictement supérieur au cardinal de F, l'existence de collision est un fait avéré. Ce qui est avancé dans la propriété de résistance aux collisions est la difficulté d'en exhiber, ne serait-ce qu'une seule.

La propriété de résistance aux collisions s'étend à des familles réduites à une seule fonction (h), c'est-à-dire dont le cardinal de l'ensemble des indices vaut 1. On dit alors simplement que la fonction h est *résistante aux collision.*

Les deux exercices qui suivent ont pour objet de proposer des construction de familles de fonctions de hachage résistantes aux collisions.

Exercice 8.1. *une famille résistante aux collisions reposant sur le problème du logarithme discret*

Soit $(G, \times)$ un groupe cyclique d'ordre premier égal à q et soit g un générateur de ce groupe. Pour tout élément y de G, on définit la fonction h_y par :

$$h_y : \begin{array}{ccc} \mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} & \to & G \\ (\alpha, \beta) & \mapsto & g^\alpha y^\beta \end{array},$$

Démontrer que si le problème du logarithme discret dans G est difficile, alors la famille de fonctions $\mathcal{H} = (h_y)_{y\in G}$ est résistante aux collisions.

Exercice 8.2. *une famille résistante aux collisions reposant sur des bijections sans pince*

Soient (f_0, f_1) une famille de bijections sans pince sur un ensemble E (voir exercice 3, page 75). Pour tout élément s de E, on définit la fonction h_s par :

$$h_s : \begin{array}{ccc} \{0,1\}^m & \to & E \\ (x_1, \ldots, x_m) & \mapsto & f_{x_m} \circ \cdots \circ f_{x_1}(s) \end{array}$$

Démontrer que la famille $\mathcal{H} = (h_s)_{s\in E}$ est résistante aux collisions.

Les fonctions résistantes aux collisions n'existent que si les fonctions à sens unique existent, la réciproque étant fausse, comme le montre l'exercice qui suit :

Exercice 8.3. a) Soit f une fonction $E \to F$ tel que tout élément de F a au moins deux antécédents. Démontrer que si f est résistante aux collisions alors f est une fonction à sens unique.
b) Démontrer que s'il existe une fonction à sens unique $E \to F$, où $\mathrm{Card}(E) > \mathrm{Card}(F)$, alors il existe une fonction à sens unique qui n'est pas résistante aux collisions.

1.2 Résistance au second antécédent

Soit $\mathcal{H} = (h_s)_{s\in\mathcal{S}}$ une famille de fonctions de hachage. De manière informelle, la résistance au second antécédent signifie qu'étant donné un élément x de E, il doit être difficile de trouver un autre élément x' qui a la même image par une fonction donnée h_s.

La possibilité de trouver un second antécédent permet bien sûr d'exhiber une collision. La résistance aux collisions implique donc la résistance au second antécédent. Cette dernière propriété est par conséquent plus plus faible que la résistance aux collision et on peut attendre que ce soit une propriété plus aisé à atteindre. Effectivement, il est possible de construire une famille résistante au second antécédent à partir d'une fonction à sens unique.

La résistance au second antécédent se modélise formellement par un jeu qui oppose un adversaire à un oracle et se quantifie par la probabilité qu'a l'adversaire à gagner à ce jeu.

Jeu 1.4. du second antécédent

1. L'adversaire choisit un élément x dans l'ensemble de départ.
2. L'oracle impose un indice $s \in \mathcal{S}$.
3. L'adversaire soit renvoyer un élément x' de E. Il a gagné si $h_s(x) = h_s(x')$ et il a perdu dans le cas contraire.

L'efficacité d'un adversaire se mesure à sa probabilité de succès à ce jeu.

Définition 1.5. famille résistante au second antécédent

On dit qu'une famille de fonctions de hachage est résistante au second antécédent si tout adversaire polynomial au jeu du second antécédent a une probabilité de succès négligeable.

Exercice 8.4. *composition des fonctions de hachage*
Soit $(h_s)_{s\in\mathcal{S}}$ une famille résistante au second antécédent et $(h'_{s'})_{s'\in\mathcal{S}'}$ une famille résistante aux collisions. Démontrer que la famille $(h_s \circ h'_{s'})_{(s,s')\in\mathcal{S}\times\mathcal{S}'}$ est résistante au second antécédent.

2 Construction d'un hachage résistant au second antécédent

L'objet de ce paragraphe est de proposer la construction d'une famille résistante au second antécédent reposant sur une fonction à sens unique. Cela démontrera que les fonctions de hachages cryptographiques ayant cette propriété appartiennent à Minicrypt.

La construction repose sur la notion de *famille universelle*.

Définition 2.1. famille universelle

Soit E un ensemble de n éléments et F un ensemble de m éléments, avec $n \geq m$. On dit que la famille $\mathcal{F} = (f_s)_{s\in\mathcal{S}}$ de fonctions $E \to F$ est une famille universelle, si pour tout couple (x, x') d'éléments distincts de E le nombre d'éléments f_s de $\mathcal{F}$ pour lesquels $f_s(x) = f_s(x')$ vaut exactement $\mathrm{Card}(\mathcal{S})/\mathrm{Card}(F)$.

Cette propriété signifie que l'application d'un élément aléatoire d'une famille universelle à un couple d'éléments distincts de E ne doit pas créer de nombre anormal de collisions. En d'autres termes, si x et x' sont deux éléments distincts de E, la probabilité que $f_s(x) = f_s(x')$, pour s tiré au hasard dans l'ensemble d'indices $\mathcal{S}$, est égale à $1/\mathrm{Card}(F)$.

Un élément d'une famille universelle est entièrement décrit par son indice, une famille sera donc d'autant plus intéressante que son cardinal est réduit. L'exercice qui suit propose des familles universelles d'intérêt croissant.

Exercice 8.5. Soient E et F deux ensembles finis tels que $\mathrm{Card}(E) \geq \mathrm{Card}(F)$.

a) Montrer que l'ensemble de toutes les fonctions $E \to F$ est une famille universelle.

b) Soient E et F des espaces vectoriels de dimension, $\dim(E) \geq \dim(F)$, sur un même corps fini $\mathbb{F}$, montrer que l'ensemble de toutes les fonctions linéaires $E \to F$ est une famille universelle.

c) Soit π une application linéaire surjective du corps $\mathbb{F}_{2^{m+1}}$ sur le corps $\mathbb{F}_{2^m}$. Montrer que la famille des fonctions $f_a : \ x \mapsto \pi(ax)$, pour $a \in \mathbb{F}^\star_{2^{m+1}}$ est universelle.

Le théorème suivant énonce que la composition d'une permutation à sens unique et d'une famille universelle conduit à une famille de fonctions de hachage résistante au second antécédent. Pour cette raison, les fonctions de hachages cryptographiques résistantes au second antécédent sont parfois appelées *Famille universelle de fonctions de hachage à sens unique*, ou *UOWHF, Universal One Way Hash Functions family.*

Théorème 2.2. Famille universelle de fonctions de hachage à sens unique deux-vers-un

Soit $(f_s)_{s\in\mathcal{S}}$ ne famille universelle de fonctions deux-vers-un $E \to F$ et soit p une permutation à sens unique sur l'ensemble E. On suppose également qu'il existe un algorithme efficace qui, pour tout couple (x, x') d'éléments distincts de E, trouve un indice s pour lequel on a $f_s(x) = f_s(x')$. Alors la famille $\mathcal{H} = (h_s)_{s\in\mathcal{S}}$, où pour tout $s \in \mathcal{S}$, on a $h_s = f_s \circ p$, est une famille de fonctions de hachage résistante au second antécédent.

Une fonction est dite *deux-vers-un* si tout élément de l'ensemble d'arrivé a exactement deux antécédents dans l'ensemble de départ. C'est le cas de la famille universelle du c) de l'exercice 5 ci-dessus. De plus, pour tout couple (x, x') d'éléments distincts de E, il existe un algorithme efficace qui permet de trouver un élément a tel que $a(x - x') \in \ker(\pi)$, donc $f_a(x) = f_a(x')$, ce qui assure la condition d'échantillonnage exigée par le théorème.

Cette construction ne permet de construire que des fonctions deux-vers-un, ce qui limite la compression à un seul symbole binaire. Il s'agit d'une première étape. Le résultat du paragraphe suivant permettra de l'étendre à des messages de taille plus importante et obtenir des taux de compression plus élevés.

Une permutation à sens unique se construit à partir de fonctions à sens unique en utilisant la construction présentée au chapitre 6 sur les chiffrements par blocs. Les fonctions de hachage résistantes au second antécédent se construisent donc à partir de fonctions à sens unique.

Preuve du théorème 2 Montrons la contraposée, c'est-à-dire si la famille $\mathcal{H}$ ne résiste pas au second antécédent, alors la permutation p n'est pas à sens unique. Supposons donc l'existence d'un adversaire A contre le second antécédent de la famille $\mathcal{H}$ qui a une probabilité de succès non négligeable, et construisons un algorithme B qui trouve un antécédent par p pour un élément $y = p(x)$, l'élément x étant aléatoire dans E. L'application p étant une permutation sur E, il est dans ce cas équivalent de choisir y aléatoire dans E.

Considérons l'algorithme B suivant, qui trouve un antécédent à un élément y de E en utilisant l'adversaire A comme sous-programme :

Algorithme 2.3.

Entrée : un élément y aléatoire de E.

1. Soit x l'élément transmis par A.
2. Calculer $z = p(x)$. Si $z = y$ alors renvoyer x. (Il ne faut pas trop compter sur un succès à cette étape)
3. Sinon, calculer l'indice s de $\mathcal{S}$ tel que $f_s(y) = f_s(z)$ et transmettre cet indice à A.
4. Soit x' le second antécédent de x par h_s proposé par A.

Sortie : Si $h_s(x') = h_s(x)$ alors renvoyer $x^\star = x'$, sinon, renvoyer une valeur $x^\star$ aléatoire.

Posons $z' = p(x')$. Le succès de A signifie que $f_s \circ p(x) = f_s \circ p(x')$, c'est-à-dire $f_s(z) = f_s(z')$. Comme la fonction f_s est supposée deux-vers-un, et comme $z' \neq z$ en raison du test à la deuxième étape, on a nécessairement $z' = y$, ce qui signifie que x' est un antécédent de y pour p. Cela signifie le succès de l'algorithme B.

Le succès de A implique donc le succès de B, donc $\mathrm{Pr}_{\mathrm{succes}}(A) \leq \mathrm{Pr}_{\mathrm{succes}}(B)$. Si le premier membre est non négligeable, alors le second ne l'est pas non plus, ce qu'il fallait démontrer. □

L'exercice suivant montre que la résistance aux collisions est une notion strictement plus forte que la résistance au second antécédent. Si les fonctions qui satisfont ce deuxième critère appartiennent à Minicrypt, savoir si celles qui sont résistantes aux collisions appartiennent aussi à Minicrypt reste un problème ouvert.

Exercice 8.6. On suppose qu'il existe des fonctions à sens unique. Construire une famille de fonctions de hachage résistante au second antécédent, mais non résistante aux collisions.

3 Hachage de messages de taille plus importante

Les familles de fonctions de hachage résistantes au second antécédent présentées au paragraphe précédent ne compressent qu'un seul symbole binaire et sont insuffisantes pour les applications pratiques.

Ce paragraphe présente la méthode de Merkle-Damgård, qui permet, à partir d'une fonction dont la compression n'est que d'un seul symbole binaire, de construire des fonctions qui acceptent des messages binaires dont la taille m est bornée par un polynôme du paramètre de sécurité, tout en conservant la propriété de résistance au second antécédent. L'application de cette méthode à la construction du paragraphe précédent permet de construire une fonction de hachage cryptographique résistante au second antécédent qui opère sur des messages de taille presque quelconque.

Théorème 3.1. construction de Merkle-Damgård

Soit $\mathcal{H} = (h_s)_{s\in\mathcal{S}}$ une famille de fonctions de hachage $E \times F \to F$. Soit m un entier borné par un polynôme du paramètre de sécurité. Pour tout m–uplet d'indices $s_1 \cdots s_m \in \mathcal{S}$, soit $h^\star_{s_1\cdots s_m}$ la fonction :

$$h^\star_{s_1\cdots s_m} : \begin{array}{ccc} E^m \times F & \to & F \\ (x_1, \ldots, x_m, r) & \mapsto & h_{s_m}(x_m, h_{s_{m-1}}(x_{m-1}, \cdots h_{s_1}(x_1, r)\cdots)) \end{array}, \tag{8.1}$$

Si la famille $\mathcal{H}$ est résistante au second antécédent, alors la famille $\mathcal{H}^\star = (h^\star_{s_1\cdots s_m})_{s_1\cdots s_m \in \mathcal{S}^m}$ est résistante au second antécédent.

Une autre façon de présenter la construction de la fonction $h^\star_{s_1\cdots s_m}$ de l'équation 8.1 est de considérer qu'elle est le résultat d'un automate dont l'état interne r_i appartient à l'ensemble F et est initialisé à $r_0 = r$. À chaque itération, pour $i = 1$ jusqu'à m, la fonction h_{s_i} met à jour

l'état interne avec l'entrée x_i par la dynamique :

$$r_i = h_{s_i}(x_i, r_{i-1})$$

Chaque itération utilise un indice s_i qui lui est propre.

Preuve Supposons l'existence d'un adversaire A contre le second antécédent pour la famille $\mathcal{H}^\star$ et montrons qu'on peut en déduire un adversaire B contre la famille $\mathcal{H}$. L'algorithme B simule l'oracle du jeu du second antécédent pour la famille $\mathcal{H}^\star$ auprès de l'adversaire A. Il joue lui-même contre un oracle du jeu du second antécédent pour la famille $\mathcal{H}$. Soit B l'algorithme suivant :

Algorithme 3.2.

1. B reçoit de A un message $x = x_1 \cdots x_m r$.
2. B choisit aléatoirement un indice $i^\star \in \{1, \ldots, m\}$.
3. B génère des valeurs aléatoires s_i, $i \neq i^\star$ aléatoirement dans $\mathcal{S}$ pour les autres indices.
4. B calcule l'entrée $r_{i^\star}$ de l'itération numéro $i^\star$. Il fournit le message $x_{i^\star} r_{i^\star}$ à son propre oracle.
5. L'oracle lui transmet alors un indice $s \in \mathcal{S}$.
6. B utilise cet indice pour l'itération $i^\star$ en posant $s_{i^\star} = s$. Il fournit alors la suite d'indices $s_1 \cdots s_m$ à l'adversaire A.
7. Soit $x' = x'_1 \cdots x'_m r'$ le second antécédent proposé par A qu'on peut supposer différent de x.
8. Si A a gagné, c'est-à-dire si $h^\star_{s_1 \cdots x_m}(x) = h^\star_{s_1 \cdots s_m}(x')$, alors, il existe nécessairement un indice $i \in \{1, \ldots, m\}$ pour lequel on a $x_i r_i \neq x'_i r'_i$ et $h_{s_i}(x_i, r_i = h_{s_i}(x'_i, r'_i)$. Dans ce cas renvoyer $x'_i r'_i$.
9. Dans le cas contraire, renvoyer une valeur $x'r'$ aléatoire.

Montrons que cet algorithme B a une probabilité de succès non négligeable. Comme l'indice $i^\star$ a été choisi au hasard, la probabilité qu'il soit égal à l'indice i défini à l'étape 8 vaut $1/m$.

D'autre part :

$$\mathrm{Pr}_{\mathrm{succes}}(B) \geq \Pr(A \text{ gagne et } i = i^\star).$$

Quelle que soit la valeur $i^\star$ choisie par B, le jeu reste équivalent pour A. Sa probabilité de succès est donc indépendante du choix de $i^\star$ par B. Donc :

$$\mathrm{Pr}_{\mathrm{succes}}(B) \geq \mathrm{Pr}_{\mathrm{succes}}(A) \times \Pr(i = i^\star) = \frac{\mathrm{Pr}_{\mathrm{succes}}(A)}{m}.$$

Si la probabilité de succès de A est non négligeable et si l'entier m est majoré par un polynôme du paramètre de sécurité, alors la probabilité de succès de B reste non négligeable. □

La construction du théorème 1 s'applique à des messages de taille m arbitraire, mais fixe. Si on accepte d'appliquer cette construction à des messages de taille variable, il est possible de construire un message qui a la même empreinte qu'un message donné. Pour pouvoir faire varier la taille des messages, une solution est de réserver le paramètre r au codage du nombre de blocs du message.

Exercice 8.7. Si on autorise des messages de taille variable dans la construction de Merkle-Damgård, montrer comment construire un second antécédent à un message $x_1 \cdots x_m r$ donné.

La construction de Merkle-Damgård s'applique également pour étendre la résistance aux collisions. Dans ce cas, la construction est plus simple, puisque le même indice s peut être utilisé à chaque itération et la fonction de hachage choisie une fois pour toute. C'est ce modèle qui est utilisé en pratique où les fonctions de hachage sont fixées et standardisées.

Exercice 8.8. *Construction de Merkle-Damgård pour des fonctions de hachage résistantes aux collisions*
On suppose que $\mathcal{H} = (h_s)_{s\in\mathcal{S}}$ une famille de fonctions de hachage $F \times E \to F$. Pour tout indice $s \in \mathcal{S}$, on définit la fonction $h_s^\star$ par :

$$h_s^\star : \begin{array}{ccc} E^\star & \to & F \\ (x_1, \ldots, x_\ell) & \mapsto & h_s\big(\cdots h_s(h_s(\ell, x_1), x_2)\cdots, x_\ell\big) \end{array},$$

où ℓ est un codage dans F de la longueur du message qui est supposée bornée par un polynôme du paramètre de sécurité. Démontrer que si la famille $\mathcal{H}$ est résistante aux collisions, alors la famille $\mathcal{H}^\star = (h_s^\star)_{s\in\mathcal{S}}$ est résistante aux collisions.

C'est cette construction qui est utilisée comme architecture générale des fonctions de hachage cryptographiques SHA1 et MD5, avec une fonction de tour h_s assez simple pour se calculer efficacement, mais assez complexe pour assurer en pratique la résistance aux collisions de la fonction composée.

4 Le paradigme « Hache-puis-signe »

Pour achever la démonstration que la signature à clé publique appartient à Minicrypt, il reste à démontrer que la composition d'une signature sûre et d'une fonction de hachage résistante au second antécédent conduit à un système de signature sûre.

Soit $\mathcal{M}$ l'ensemble des messages et soit $\mathcal{H} = (h_s)_{s\in\mathcal{S}}$ une famille de fonctions de hachage $\mathcal{M} \to E$ dont l'ensemble d'indice est $\mathcal{S}$.

Soit Σ un système de signature dont l'ensemble des messages est le produit $E \times \mathcal{S}$ de l'ensemble E des valeurs de hachage et de l'ensemble $\mathcal{S}$ des indices de la famille de fonctions de hachage.

On considère le système de signature $\widehat{\Sigma}$ qui consiste à composer une fonction h_s de la famille $\mathcal{H}$ avec le système de signature Σ.

Plus précisément :

— Les clés publiques et privées de $\widehat{\Sigma}$ sont les mêmes que les clés publiques et privées de Σ.

— Pour signer un message $m \in \mathcal{M}$:

1. Générer un indice s dans $\mathcal{S}$ pour la famille de fonctions de hachage ;
2. Calculer l'empreinte $h = h_s(m)$ du message m ;
3. Signer dans le système Σ le couple (s, h). Soit σ la signature.
4. La signature de m dans le système $\widehat{\Sigma}$ est le couple (s, σ).

— Pour vérifier la signature (s, σ) d'un message m :

1. Calculer $h_s(m)$;
2. Vérifier dans le système Σ que la signature σ est bien celle du couple $(s, h_s(m))$.

Théorème 4.1.

Si le système de signature Σ est sûr contre une attaque à messages choisis et si la famille $\mathcal{H}$ de fonctions de hachage est résistante au second antécédent, alors le système de signature $\widehat{\Sigma}$ est sûr contre une attaque à messages choisis.

Preuve Supposons la famille $\mathcal{H}$ résistante au second antécédent.

Supposons l'existence d'un adversaire $\widehat{A}$ contre le système de signature $\widehat{\Sigma}$ et construisons un adversaire A contre le système Σ et son objectif est de fournir un message signé dans le système Σ.

L'adversaire A simule le rôle d'oracle pour la signature $\widehat{\Sigma}$ auprès de $\widehat{A}$. Il joue lui-même contre un oracle pour le système de signature Σ.

Considérons l'algorithme A suivant :

Algorithme 4.2.

- Lorsque $\widehat{A}$ requiert la signature d'un message choisi m_i, alors :
  - il génère un indice s_i,
  - il calcule l'image $h_i = h_{s_i}(mi)$,
  - il transmet le couple (s_i, h_i) à l'oracle du système Σ, soit σ_i la signature rendue.
  - il transmet à $\widehat{A}$ la signature (s_i, σ_i) du message m_i.
- L'adversaire $\widehat{A}$ renvoie alors le message m avec sa signature forgée $\widehat{\sigma} = (s, \sigma)$.
- A calcule $h = h_s(m)$ et renvoie le message (s, h) avec la signature forgée σ.

Il reste à évaluer la probabilité de succès de cet adversaire A contre le système Σ.

Si l'adversaire $\widehat{A}$ a réussi à forger une signature valide pour le message m, alors l'algorithme A a également réussi à forger une signature valide pour le message (s, h). Mais il n'a gagné au jeu de l'authentification que si le message (s, h) renvoyé par A est différent de tous les messages (s_i, h_i) requis au cours du déroulement du jeu. Notons E l'événement « *Il existe un indice i tel que le message requis (s_i, h_i) auprès de l'oracle est égal au message (s, h) renvoyé par l'adversaire A.* ».

Si l'événement E est survenu, c'est-à-dire si (s, h) est égal à l'un des (s_i, h_i), alors on peut dire que $\widehat{A}$ a gagné au jeu du second antécédent contre la famille de fonctions de hachage $\mathcal{H}$. En effet :

- Il a fourni un message m_i.
- Au travers de la signature (s_i, h_i), l'algorithme A a ensuite imposé à $\widehat{A}$ l'indice s_i.
- Et finalement, $\widehat{A}$ a exhibé un second antécédent m pour la fonction de hachage d'indice s_i, puisque $h_{s_i}(m_i) = h_s(m) = h_s(m_i)$.

Comme on a supposé que la famille $\mathcal{H}$ est résistante au second antécédent, l'événement E a une probabilité négligeable.

Le succès de $\widehat{A}$ est la réunion disjointe des événements « succès de $\widehat{A}$ et E » et « succès de $\widehat{A}$ et $\overline{E}$ » :

$$\mathrm{Pr}_{\mathrm{succes}}(\widehat{A}) = \underbrace{\mathrm{Pr}(\mathrm{succes}(A) \text{ et } \overline{E})}_{\leq \mathrm{Pr}_{\mathrm{succes}}(A)} + \underbrace{\mathrm{Pr}(\mathrm{succes}(A) \text{ et } E)}_{\leq \mathrm{Pr}(E)}$$

Il en résulte :

$$\mathrm{Pr}_{\mathrm{succes}}(A) \geq \mathrm{Pr}_{\mathrm{success}}(\widehat{A}) - \mathrm{Pr}(E)$$

La probabilité de succès de A est donc non négligeable, comme supérieure à la différence d'une quantité non négligeable et d'une quantité négligeable, ce qui achève la preuve. □

Cette signature requiert d'ajouter l'indice s au message à signer, ce qui peut poser des problèmes pratiques lorsque cet indice est grand. Or dans la construction de Merkle-Damgård, cet indice est la concaténation de m indices choisis indépendamment, ce qui est effectivement une donnée de grande taille. Le problème se résout en remplaçant s par son empreinte par une fonction de hachage choisie une fois pour toute.

Si la famille de fonctions de hachage a la propriété plus forte de résistance aux collisions, la définition d'un système de signature *hache-puis-signe* est plus simple. Il suffit dans ce cas de définir un indice de fonction de hachage une fois pour toutes au lieu de le changer à chaque signature comme c'est le cas lors de l'application du théorème 1. C'est ainsi que fonctionnent en pratique les algorithmes de signature du monde réel.

L'objet de l'exercice qui suit est de prouver la sécurité du système de signature suivant le paradigme hache-puis-signe lorsque la famille de fonctions de hachage est résistante aux collisions et que l'indice de la fonction de hachage est choisi une fois pour toutes, comme cela est pratiqué dans le monde de la cryptographie réelle.

Exercice 8.9. Soit Σ un système de signature opérant sur un ensemble fini E et soit $\mathcal{H} = (h_s)_{s\in\mathcal{S}}$ une famille de fonctions de hachage $\mathcal{M} \to E$, dont l'ensemble d'indices est $\mathcal{S}$. Soit $\widehat{\Sigma}$ le système de signature défini sur l'ensemble $\mathcal{M}$ des messages comme suit :

- Une clé publique de $\widehat{\Sigma}$ est un couple (k_{pub}, s) où k_{pub} est une clé publique de Σ et s un indice de $\mathcal{S}$.
- La clé privée de $\widehat{\Sigma}$ est la clé privée correspondant à k_{pum} dans le système Σ.
- La signature du message m pour $\widehat{\Sigma}$ est la signature du message $h_s(m)$ pour Σ.
- Pour vérifier la signature σ du message m pour le système $\widehat{\Sigma}$, on vérifie la signature du message $h_s(m)$ pour le système Σ.

Démontrer que si Σ est sûr contre une attaque à messages choisis et si la famille $\mathcal{H}$ est résistante aux collisions, alors le système $\widehat{\Sigma}$ est sûr contre une attaque à messages choisis.

9 Chiffrement à clé publique

Un système de confidentialité est dit *à clé publique* si d'une part les clés de chiffrement et de déchiffrement sont différentes, et si d'autre part la clé de déchiffrement ne peut pas se déduire efficacement de la clé de chiffrement. Il n'y a donc aucun inconvénient à publier la clé de chiffrement, ce qui évite d'avoir à communiquer préalablement une clé secrète à son correspondant. Les clés de chiffrement de chaque destinataire peuvent alors figurer dans un annuaire public, ce qui permet à tous de chiffrer un message pour qui a publié sa clé publique. Le secret n'a pas besoin de voyager, il reste localisé au déchiffrement. La gestion des clés s'en trouve grandement simplifiée et c'est précisément pour résoudre ce problème que le chiffrement à clé publique a été développé dans les années 1970, d'abord en Angleterre, puis aux États-Unis d'Amérique.

Le chiffrement à clé publique repose sur la notion de *fonction à sens unique avec trappe* (*One way trapdoor function*). Ce sont des fonctions difficiles à inverser sauf pour qui détient une information particulière supplémentaire, appelé la *trappe*, et qui permet une inversion efficace. Ce type de fonction n'existe pas dans Minicrypt, ce qui inscrit le chiffrement à clé publique dans le monde Cryptomania.

En pratique, le chiffrement à clé publique est plus lent qu'un chiffrement symétrique avec des algorithmes standards comme le DES ou l'AES. Il n'est pas utilisé pour chiffrer de longs messages, mais seulement pour chiffrer une clé de session qui sera utilisée pour chiffrer le message lui-même à l'aide d'un algorithme symétrique. Il n'y a donc pas d'inconvénient à ce qu'un système de chiffrement à clé publique opère sur un espace des messages dont la taille est limité à celle de la clé d'un chiffrement symétrique, soit quelques centaines de symboles binaires.

1 La sécurité du chiffrement à clé publique

La clé de chiffrement étant publique, elle est connue de tous et en particulier de l'adversaire. Celui-ci peut donc chiffrer les messages de son choix. Il a donc naturellement accès à un oracle de chiffrement. Le modèle minimal est donc celui de l'attaque à clairs choisis CPA *Chosen Plaintext Attack*.

Cette notion est considérée comme insuffisante, et les systèmes de chiffrement à clé publique doivent également résister à des attaques au cours desquelles l'adversaire a accès à un oracle de déchiffrement. Il s'agit là d'attaques à cryptogrammes choisis CCA *Chosen Ciphertext Attack*.

Le premier procédé de chiffrement à clé publique à avoir été publié est le RSA. L'opération de chiffrement, $\mathrm{RSA}_e : \ m \mapsto m^e \bmod n$ est une opération inversible calculable par tous, mais l'opération inverse n'est accessible qu'à ceux qui ont la connaissance de la factorisation du module n, cette connaissance faisant office de trappe. Toutefois, le lien entre connaissance de la factorisation du module et capacité à inverser la fonction RSA_e n'est toujours pas établi. Il n'existe pas de preuve que la factorisation de n soit nécessaire pour inverser la fonction RSA.

Le premier procédé à avoir été assorti d'un argument de sécurité est le système de Rabin. Le chiffrement consiste à élever le message en clair au carré modulo un entier n égal au produit de deux grands nombres premiers distincts. L'équivalence entre capacité à calculer les racines carrées et connaissance de la factorisation de n est établie (voir exercice 6, page 11), ce qui constitue un argument de sécurité : tant que la factorisation des entiers sera un problème difficile, il sera également difficile de décrypter les messages chiffrés ainsi.

Mais cet argument de sécurité s'effondre lorsqu'on considère des attaques à cryptogramme choisi. Si un adversaire a accès à un oracle de déchiffrement et peut faire déchiffrer les messages de son choix, il peut alors factoriser le module et rendre inopérante toute protection.

Par ailleurs, il n'est pas nécessaire de décrypter entièrement les messages pour considérer un mécanisme de chiffrement comme non sûr. Un système de confidentialité doit protéger tous les bits d'information du message en clair, ce qui est modélisé par la notion de sécurité sémantique. Cette notion est équivalente à celle d'indistinguabilité, évaluée par un jeu au cours duquel l'adversaire choisit deux messages, l'oracle chiffre l'un des deux, renvoie un cryptogramme, et l'adversaire doit retrouver quel est le message qui a été chiffré.

On distingue alors deux niveaux d'attaques à cryptogramme choisis. Si l'adversaire n'a accès à l'oracle de déchiffrement qu'avant avoir reçu le cryptogramme à résoudre, on parle d'attaque CCA1. Si au contraire l'adversaire a accès à l'oracle de déchiffrement pendant tout le jeu, on parle d'attaque CCA2, ce qui est le niveau standard de sécurité requis.

Pour qu'un chiffrement ait la propriété d'indistinguabilité, il est nécessaire qu'il soit non déterministe. Il faut donc inclure de l'aléa dans le procédé de chiffrement. Une idée est donc de compléter le message en clair avec un élément aléatoire (*padding* = remplissage) avant de lui appliquer la fonction de chiffrement. La version 1.5 du standard de chiffrement à clé publique PKCS#1, publiée par l'entreprise *RSA laboratories* en 1993 a défini un modèle de remplissage aléattoire pour le RSA. Ce standard n'a résisté que cinq années. En 1998, Daniel Bleichenbacher a publié une méthode qui permet à un adversaire de décrypter n'importe quel message par une attaque effective à cryptogrammes choisis, en tirant partie de l'information sur l'échec du déchiffrement.

La situation extrême pour la fonction RSA est de ne chiffrer qu'un seul symbole binaire et de remplir avec des symboles aléatoires. L'exercice 13, page 18 sur la sécurité du chiffre de parité de la fonction RSA montre qu'un tel procédé de chiffrement est IND-CPA. On ne peut malheureusement pas en dire plus. Un tel chiffrement ne résiste pas à une attaque à cryptogramme choisi, ce que l'exercice suivant propose de montrer.

Exercice 9.1. Montrer que le chiffrement RSA d'un seul bit avec remplissage aléatoire n'est pas IND-CCA.

Pour résister aux attaques à cryptogrammes choisis, les cryptologues Mihir Bellare et Philip Rogaway ont proposé un mode d'utilisation pour n'importe quel chiffrement à clé publique, appelé OAEP (*Optimal Asymetric Encryption Padding*). Ce mode d'utilisation consiste à faire passer le message en clair dans deux tours d'un schéma de Feistel dans lequel les fonction de tour sont des oracles aléatoires. Bellare et Rogaway avaient assorti leur proposition d'une preuve de sécurité. Ce mode d'utilisation a été standardisé pour la sécurité de l'Internet dans les normes ANSI X9-44, IEEE P1363 et SET (*Secure Electronic Transaction*). Il a été également adopté dans la norme EMV (*Europay, Mastercard, Visa*) des cartes bancaires, conjointement avec le RSA, puis intégré à la version 2 de la norme PKCS#1 en 1998.

Mais en 2001, Victor Shoup a découvert une faille dans la preuve de sécurité et a même présenté une attaque avec un chiffrement à clé publique particulier. Pour corriger la preuve, Shoup a ajouté une hypothèse sur la fonction de chiffrement utilisée : elle doit être « *à sens unique séparable* », en laissant ouverte la question de savoir si le RSA satisfait ou non cette nouvelle hypothèse.

En 2003, Phan et Pointcheval proposent une modification du mode d'utilisation – avec trois tours du schéma de Feistel au lieu de deux – qui permet de se dispenser de cette hypothèse supplémentaire, mais il était trop tard pour faire évoluer le standard.

Finalement, en 2004, Fujisaki, Okamoto, Pointcheval et Stern prouvent que le RSA satisfait l'hypothèse de séparabilité nécessaire pour prouver la sécurité du remplissage OAEP avec la fonction RSA, rassurant définitivement sur la sécurité du standard.

Toutefois, le recours à des oracles aléatoires est jugé par certains théoriciens de la cryptographie comme une hypothèse trop forte pour assurer une sécurité en pratique, ce qui a conduit à l'élaboration d'un système de chiffrement qui se dispense de cet usage et qui se contente de simples fonctions de hachages. L'objet de la suite de ce chapitre est de présenter les outils et les problèmes à trappe qui permettent de modéliser la sécurité du chiffrement à clé publique et de présenter le système de chiffrement de Cramer-Shoup, sûr contre une attaque à cryptogrammes choisis dans un modèle réaliste de résistante aux collisions.

2 Problèmes avec trappe

Le chiffrement à clé publique reposant sur la notion de fonction à sens unique avec trappe, il convient de préciser quels sont aujourd'hui les problèmes qui sont supposés avoir cette propriété. Commençons par les définir.

Définition 2.1. problème avec trappe

Un problème avec trappe est un problème difficile qui devient facile lorsqu'on dispose d'une information supplémentaire qui permet de le résoudre avec un algorithme efficace.

Un problème difficile étant un problème pour lequel aucun algorithme efficace n'a de probabilité de succès non négligeable, la notion de problème avec trappe est une notion asymptotique. Il s'agit donc d'une famille de problèmes indexée par un paramètre de sécurité.

Le problèmes avec trappe se range dans deux catégorie : les problèmes calculatoires, pour lesquels le calcul d'un résultat est attendu, et les problèmes décisionnels pour lesquels la réponse est une réponse binaire *oui*-ou-*non*. Les premiers sont *a priori* plus difficiles que les seconds, mais certains problèmes décisionnels résistent toujours et les preuves de sécurité des mécanismes de chiffrement ne sont établies que sous l'hypothèse de la difficulté du problème décisionnel.

Quelques problèmes calculatoires à trappe

La racine carrée modulo un entier composite

Étant donné un entier composite $n = p \times q$, le calcul du carré modulo n est facile, mais étant donné y qui est un carré dans $\mathbb{Z}/n\mathbb{Z}$, trouver une racine carrée est difficile, sauf lorsque la factorisation de n est connue. Voir exercice 6, page 11.

L'inversion de la fonction RSA

Étant donné un entier $n = p \times q$ égal au produit de deux nombres premiers impairs distincts et un exposant e premier avec $p-1$ et $q-1$, l'application $x \mapsto x^e$ est une bijection de $\mathbb{Z}/n\mathbb{Z}$. Il est facile pour tout $x \in \mathbb{Z}/n\mathbb{Z}$ de calculer $x^e \bmod n$. Le calcul de la bijection inverse est reconnu comme difficile sauf lorsque les facteurs p et q sont connus.

Lorsque les facteurs p et q sont connus, une façon d'inverser la fonction RSA est l'élévation à la puissance d, où d, appelé *exposant privé*, est l'inverse de e modulo $\lambda(n) = \text{ppcm}(p-1, q-1)$. Mais il n'est toujours pas établi que la factorisation de l'entier n soit nécessaire pour inverser cette fonction et qu'un autre procédé que l'élévation à la puissance d soit envisageable.

Par contre, la connaissance de l'exposant privé permet la factorisation du module :

Exercice 9.2. Démontrer que la connaissance de l'exposant privé permet de factoriser l'entier n de manière efficace.

Le problème Diffie-Hellman calculatoire

Soit $(G, \times)$ un groupe cyclique et g un générateur. Le problème Diffie-Hellman consiste, étant donné x et y dans G, à trouver l'élément $z = g^{ab}$, où $a = \log_g(x)$ et $b = \log_g(y)$. Ce problème est issu du mécanisme d'échange de clés que Diffie et Hellman ont publié dans leur article de 1976 [3]. L'élément z est la clé commune à deux entités qui ont échangé x et y sur un canal public. Ce problème reste difficile dans le groupe multiplicatifs des entiers modulo un nombre premier p assez grand. Il l'est aussi dans d'autres groupes comme le groupe des points d'une courbe elliptique sur un corps fini. Mais lorsque $\log_g(x) = a$ est connu, la solution $z = y^a$ est accessible.

Le problème Diffie-Hellman calculatoire est bien sûr plus facile que le calcul du logarithme discret en base g dans le groupe $(G, \times)$. Mais on soupçonne, sans qu'on sache le démontrer qu'il est en fait de difficulté équivalente.

Problèmes décisionnels à trappe

Le problème de résiduosité quadratique

Soit n un entier égal au produit de deux nombres premiers distincts p et q. Étant donné un élément x de $\mathbb{Z}/n\mathbb{Z}$, le problème de résiduosité quadratique est celui de répondre à la question :« l'élément x est-il ou non égal au carré d'un élément y de $\mathbb{Z}/n\mathbb{Z}$. Le théorème chinois des restes exprime que x est un carré modulo n si et seulement si il est à la fois un carré modulo p et un carré modulo q, c'est-à-dire si les symboles de Legendre $\left(\frac{x}{p}\right)$ et $\left(\frac{x}{q}\right)$ sont tous deux égaux à 1. Le symbole de Jacobi $\left(\frac{x}{n}\right)$ étant le produit de ces deux symboles de Legendre, il vaut nécessairement 1 si x est un carré modulo n.

Lorsque le symbole de Jacobi $\left(\frac{x}{n}\right)$ vaut -1, alors il est le produit $1 \times (-1)$ ou $(-1) \times 1$, et x n'est certainement pas un carré modulo n. Mais si sa valeur est 1, cela peut être le résultat de 1×1 ou de $(1) \times (-1)$, et il reste difficile de déterminer si x est ou non un carré modulo n en l'absence de la connaissance de la factorisation de n.

Par contre, si les facteurs p et q de n sont connus, savoir si x est un carré modulo n se déduit directement du calcul des symboles de Legendre $\left(\frac{x}{p}\right)$ et $\left(\frac{x}{q}\right)$.

Le problème de résiduosité quadratique s'exprime également en terme d'appartenance à un sous-groupe. Soit G le sous-groupe du groupe multiplicatif des éléments inversibles modulo n. Ce dernier est défini par :

$$G = \left\{ x \in \mathbb{Z}/n\mathbb{Z}^\star \mid \left(\frac{n}{x}\right) = 1 \right\}$$

Soit H le sous-groupe de G défini par :

$$H = \{x \in G \mid \exists y \in \mathbb{Z}/n\mathbb{Z}^\star,\ x = y^2\}.$$

Le problème de résiduosité quadratique peut se reformuler ainsi : « l'élément x de G appartient-il au sous-groupe H ? »

On se limite aux éléments inversibles, car si l'élément x est non nul et non inversible modulo n, c'est qu'il n'est pas premier avec n. Dans ce cas, l'algorithme d'Euclide révèle la factorisation de n. Le problème de résiduosité quadratique devient facile.

Le problème Diffie-Hellman décisionnel

Soit $(G, \times)$ un groupe cyclique d'ordre q premier et soit g un générateur de G. On dit que le triplet (x, y, z) est un triplet Diffie-Hellman s'il existe deux entiers a et b tels que $x = g^a$, $y = g^b$ et $z = g^{ab}$. Les entiers a et b sont respectivement les logarithmes en base g de x et y. Le problème Diffie-Hellman décisionnel est de savoir dire si un triplet (x, y, z) est ou non un triplet Diffie-Hellman. En d'autres termes, cela revient à demander si z est, ou n'est pas, la clé convenue entre deux entités qui ont échangé les données x et y sur un canal public.

Si on sait résoudre le problème Diffie-Hellman calculatoire, alors on sait aussi résoudre le problème Diffie-Hellman décisionnel. Ce dernier est donc *a priori* plus facile que le problème calculatoire. Mais la question de la réciproque, probablement fausse, reste ouverte.

Le problème Diffie-Hellman décisionnel peut aussi s'exprimer comme un problème d'appartenance à un sous-groupe.

Définition 2.2. sous-groupe Diffie-Hellman

Soit $(G, \times)$ un groupe cyclique d'ordre q premier et soit g un générateur. Pour tout élément y de G, on appelle sous-groupe Diffie-Hellman le sous-groupe H_y du groupe produit $G \times G$ engendré par le couple (g, y), soit :

$$H_y = \{(g^r, y^r) \mid r \in \mathbb{Z}/q\mathbb{Z}\}.$$

L'introduction de cette définition permet de reformuler le problème Diffie-Hellman par la proposition suivante dont la preuve est directe.

Proposition 2.3.

Le triplet (x, y, z) est un triplet Diffie-Hellman si et seulement si le couple (x, z) appartient au sous-groupe Diffie-Hellman H_y.

Le problème des classes de degré composite

Soit n un entier égal au produit de deux nombres premiers distincts $n = p \times q$, tel que n est premier avec $p - 1$ et $q - 1$. Le problème de classe de degré composé consiste à répondre à la question suivante : « l'entier y de $\mathbb{Z}/n^2\mathbb{Z}$ est-il la puissance n-ième d'un élément r de $\mathbb{Z}/n\mathbb{Z}$. »

Ce problème est considéré comme difficile si la factorisation de n n'est pas connu. Il est cependant facile si les facteurs p et q sont connus.

Exercice 9.3. On suppose que $n = p \times q$ avec p et q connus. Soit y un élément de $\mathbb{Z}/n^2\mathbb{Z}$. Comment déterminer s'il existe un élément $r \in \mathbb{Z}/n\mathbb{Z}$ tel que $y = r^n \bmod n^2$?
Indication : Poser $r^\star = a^d \bmod n^2$, où d est l'inverse de n modulo $\lambda(n) = \text{ppcm}(p - 1, q - 1)$.

Comme les problèmes précédents, le problème de la classe de degrés composés peut s'exprimer comme un problème d'appartenance à un sous groupe : L'élément y de $\mathbb{Z}/n^2\mathbb{Z}^\star$ appartient-il au sous-groupe H de $\mathbb{Z}/n^2\mathbb{Z}^\star$ défini par :

$$H = \{r^n \in \mathbb{Z}/n^2\mathbb{Z} \mid r \in \mathbb{Z}/n\mathbb{Z}^\star\}?$$

3 Problème de l'appartenance à un sous-groupe

Les problèmes décisionnels présentés dans le paragraphe précédent peuvent tous s'exprimer comme problème avec trappe d'appartenance à un sous-groupe. Cela signifie qu'il est difficile de décider de l'appartenance ou non d'un élément d'un groupe au sous-groupe, et que cette difficulté disparait lorsqu'une information supplémentaire est disponible. Dans les problèmes présentés, cette information est dans certains cas la connaissance du logarithme d'un élément, et dans d'autres la connaissance de la factorisation d'un grand entier.

L'objet de ce paragraphe est de formaliser ce type de problème et de présenter des systèmes de chiffrement à clé publique qui reposent sur la difficulté ces problèmes.

L'analyse de la sécurité étant un problème asymptotique, on doit considérer une famille de groupes $(G, \times)$ indexée par un paramètre de sécurité. S'agissant par exemple du groupe $(\mathbb{Z}/p\mathbb{Z}^\star, \times)$ où p est un nombre premier, le paramètre de sécurité est le nombre de chiffres binaires de l'entier p.

Soit également H un sous-groupe de G.

On suppose par ailleurs que :

— Il existe des algorithmes efficaces qui réalisent les opérations de groupe – multiplication et inversion.

— Il existe deux algorithmes efficaces pour générer des éléments aléatoires de G et de H.

La difficulté de décider de l'appartenance d'un élément x de G au sous-groupe H s'évalue par la capacité qu'a un adversaire à gagner à un jeu qui consiste à dire si oui ou non des éléments qui lui sont présentés appartient au groupe G ou au sous-groupe H.

Ce jeu oppose l'adversaire à un oracle et est formellement défini ainsi :

Jeu 3.1. d'appartenance à un sous-groupe

1. L'oracle choisit aléatoirement un symbole binaire $b \in_R \{0,1\}$.
2. Si $b = 0$, l'oracle génère un nombre m d'éléments aléatoires x dans G, et si $b = 1$, il génère m éléments aléatoires x dans H, l'entier m étant borné par un polynôme du paramètre de sécurité. Il transmet ces éléments à l'adversaire.
3. Au bout d'un temps borné par un polynôme du paramètre de sécurité, l'adversaire renvoie une estimation $b^\star$ de b.

L'adversaire a gagné si $b = b^\star$, il a perdu dans le cas contraire. Son efficacité se mesure à son avantage. Cela conduit à la définition suivante :

Définition 3.2. sous-groupe inextricable

On dit que le sous-groupe H de G est inextricable (*intractable*) si l'avantage de tout adversaire polynomial au jeu d'appartenance à ce sous-groupe est une fonction négligeable du paramètre de sécurité.

On admettra que les sous-groupes présentés au paragraphe précédent pour modéliser les problèmes décisionnels à trappe – sous-groupe des carrés modulo un entier composite, sous-groupe Diffie-Hellman et sous-groupe de classe de degré composite – sont inextricables.

Exercice 9.4. On suppose qu'à la deuxième étape du jeu 1, l'oracle génère et transmet un seul élément au lieu d'un nombre m. On note alors ce jeu, le *jeu-1*. Le jeu 1 peut donc être noté *jeu-m*. Démontrer qu'un sous-groupe H de G est inextricable pour le jeu-m si et seulement si il est inextricable pour le jeu-1.
Indication : s'inspirer de la preuve du théorème 2, page 29.

On pourra donc supposer que l'oracle fournit à l'adversaire une ou plusieurs observations du groupe ou du sous-groupe.

Au cours du jeu d'appartenance à un sous-groupe présenté ci-dessus, il se peut que l'oracle choisisse $b = 0$, tire au hasard un ou plusieurs éléments x dans G et que ces éléments appartiennent fortuitement au sous-groupe H. Un distingueur ne peut en fait pas faire autrement que de prendre sa décision sur la base de l'appartenance ou non de x au sous-groupe H. Il est donc naturel de définir une variante du jeu d'appartenance, un jeu*, identique au jeu 1 ci-dessus, mais au cours duquel si $b = 0$, l'oracle tire explicitement un élément aléatoire dans $G \setminus H$ qui n'appartient pas à H au lieu de le tirer au hasard dans G. Ces deux jeux sont en fait équivalents comme le montre l'exercice ci-après. Un adversaire au jeu du sous-groupe est finalement un distingueur des sous-ensembles H et de $G \setminus H$. On pourra donc indifféremment faire jouer un distingueur à l'une ou l'autre variante du jeu.

Exercice 9.5. Soit D un distingueur d'un sous-groupe H de G. On note $\mathrm{Av}^\star(D)$ son avantage lorsqu'il joue à la variante jeu* définie dans la remarque ci-dessus. Démontrer que $\mathrm{Av}(D) = \Pr(x \in G \setminus H) \times \mathrm{Av}^\star(D)$.

4 Chiffrement à clé publique avec des sous-groupes inextricables

Ce paragraphe décrit trois systèmes de chiffrement à clé publique assortis chacun d'une preuve de sécurité reposant sur l'inextricabilité des sous-groupes présentés plus haut.

Chiffrement de Goldwasser-Micali

Le chiffrement de Goldwasser-Micali (1982) est le premier système de chiffrement à clé publique a avoir été assorti d'une preuve de sécurité. Il repose sur l'inextricabilité du sous groupe des carrés modulo un

entier composite. Il est cependant très peu efficace et donc totalement inutilisé, puisqu'il faut aujourd'hui de l'ordre d'un millier de symboles binaires pour n'en chiffrer un seul. Une telle expansion des cryptogrammes est rédhibitoire pour toute utilisation pratique. L'intérêt de ce chiffrement est essentiellement historique.

Ce système de chiffrement est défini ainsi :

— Génération des clés : Soit n un entier égal au produit de deux nombres premiers distincts : $n = p \times q$.
  — Générer un entier e qui n'est un carré ni modulo p, ni modulo q, c'est-à-dire $\left(\frac{e}{p}\right) = -1$ et $\left(\frac{e}{q}\right) = -1$.
  — La clé publique est le couple (n, e).
  — La clé privée est le couple (p, q). Cette clé privée permet de déterminer si un entier c de $\mathbb{Z}/n\mathbb{Z}$ vérifiant $\left(\frac{c}{n}\right) = 1$ est ou n'est pas un carré modulo n.

— Chiffrement Soit $m = m_1 \cdots m_\ell \in \{0,1\}^\ell$ un message constitué de ℓ symboles binaires. Pour chaque indice $i \in \{1, \ldots, \ell\}$:
  — Générer un élément aléatoire r_i de $\mathbb{Z}/n\mathbb{Z}$.
  — Calculer $c_i = r_i^2 \times e^{m_i}$ dans $\mathbb{Z}/n\mathbb{Z}$.
  — Le cryptogramme est $c = c_1 \cdots c_\ell \in (\mathbb{Z}/n\mathbb{Z})^\ell$.

— Déchiffrement Pour chaque indice $i \in \{1, \ldots, \ell\}$, si c_i est un carré, alors $m_i = 0$, et si c_i n'est pas un carré, alors $m_i = 1$.

Chaque terme c_i du cryptogramme contient l'information binaire du message masquée par un carré aléatoire. Comme e n'est un carré ni modulo p, ni modulo q le symbole de Jacobi $\left(\frac{c_i}{n}\right)$ vaut toujours 1, ce qui n'aide pas à déterminer si c_i est ou non un carré modulo n.

Ce chiffrement a une propriété d'homomorphisme. Si c et c' sont les cryptogrammes des messages m et m', alors le cryptogramme $c \times c'$ défini par le produit terme à terme des cryptogrammes c et c' est le cryptogramme $m + m'$ défini par la somme terme à terme modulo 2 des messages m et m'. Cette propriété d'homomorphisme a pour conséquence que les cryptogrammes sont malléables. Il ne faut donc pas s'attendre à ce que ce type de chiffrement soit résistant à une attaque à cryptogramme choisi.

Exercice 9.6. Montrer que le chiffrement de Goldwasser-Micali n'est pas IND-CCA.

Par contre, ce chiffrement est assorti d'une preuve de sécurité, comme l'énonce la proposition suivante :

Proposition 4.1. IND-CPA du chiffrement de Goldwasser-Micali

Si le sous-groupe H des carrés modulo n est inextricable, alors le chiffrement de Goldwasser-Micali est IND-CPA.

Preuve Montrons la contraposée. On suppose l'existence d'un adversaire A contre l'indistinguabilité qui a un avantage non négligeable, et montrons l'existence d'un distingueur D du sous-groupe des carrés modulo n qui a un avantage non négligeable.

Le distingueur D utilise l'adversaire A comme sous-programme. On peut supposer que D joue contre l'oracle au jeu-ℓ, où ℓ est la taille des messages transmis par l'adversaire. L'oracle choisit un symbole binaire $\widehat{b} \in_R \{0,1\}$. Si $b = 0$, il choisit les termes x_i aléatoires dans le sous-groupe G de $\mathbb{Z}/n\mathbb{Z}^\star$ des éléments t dont le symbole de Jacobi $\left(\frac{n}{t}\right)$ vaut 1, et si $b = 1$ il choisit les termes x_i aléatoires dans le sous-groupe des carrés de G.

Considérons le distingueur D suivant :

Algorithme 4.2.

Entrée : une liste de ℓ termes $x_1, \ldots, x_\ell$ choisis par l'oracle.
- D reçoit de A deux messages m^0 et m^1 de ℓ symboles binaires. Il choisit $b \in_R \{0,1\}$ aléatoire et transmet le cryptogramme dont les termes sont ceux de m_b masqués par les x_i, c'est-à-dire $x_i \times e^{m_i^b}$.
- Après un temps polynomial, A propose une estimation $b^\star$ de b.

Sortie : Si $b^\star = b$ alors D renvoie $\widetilde{b} = 1$, sinon, il renvoie $\widetilde{b} = 0$.

L'heuristique qui guide la définition de ce distingueur est la suivante : si A trouve la bonne valeur de b, c'est probablement que les termes x_i sont des carrés. Il reste à évaluer l'avantage de D et à montrer qu'il n'est pas négligeable.

Par définition :

$$\begin{aligned} \mathrm{Av}(D) &= \left|\Pr(\widetilde{b} = 1 \mid \widehat{b} = 1) - \Pr(\widetilde{b} = 1 \mid \widehat{b} = 0)\right| \\ &= \left|\Pr(b^\star = b \mid x \in H) - \Pr(b^\star = b \mid x \in G)\right| \end{aligned}$$

Le premier terme de cette différence est la probabilité que A gagne alors que le cryptogramme qui lui a été transmis a été correctement construit. Il s'agit de la probabilité de succès de A.

Le deuxième terme est la probabilité que A gagne alors que le cryptogramme qui lui a été transmis est constitué de termes tous aléatoires dans G. Le cryptogramme a donc autant de chance d'être celui de m^0 que celui de m^1. La probabilité de succès de A dans ce cas vaut $1/2$. Il en résulte que :

$$\mathrm{Av}(D) = \left|\Pr_{\text{succes}}(A) - \frac{1}{2}\right| = \frac{1}{2}\,\mathrm{Av}(A),$$

qui est non négligeable par hypothèse et qui achève la preuve. □

Chiffrement ElGamal

Ce système de chiffrement à clé publique a été proposé par Taher ElGamal en 1984. Il repose sur la difficulté du logarithme discret et s'inspire directement du protocole d'échanges de clés Diffie-Hellman. Ce système utilise un groupe cyclique $(G, \times)$ d'ordre q premier engendré par g. Il est défini ainsi :

- Génération des clés :
  - La clé privée est un élément d aléatoire de $\mathbb{Z}/q\mathbb{Z}$.
  - La clé publique est $e = g^d$.
- Chiffrement : Pour chiffrer un message $m \in G$:
  - Choisir un élément aléatoire $r \in_R \mathbb{Z}/q\mathbb{Z}$ et calculer $\mu = e^r$.
  - Le cryptogramme est le couple $\gamma = (u, c)$, où $u = g^r$ et $c = \mu \times m$.
- Déchiffrement : Pour déchiffrer le cryptogramme $\gamma = (u, c)$:
  - Calculer $\mu = u^d = g^{rd} = e^r$.
  - Retrouver m par $m = c/\mu$.

Dans le cryptogramme (u, c), la composante c porte l'information du message m masquée par la donnée aléatoire $\mu = e^r$, et la composante u est un indice qui permet au détenteur de la clé privée de retrouver le masque μ.

Ce système de chiffrement dispose d'une preuve de sécurité contre une attaque à clairs choisis, sur la base de l'inextricabiité du sous-groupe Diffie-Hellman, mais pas contre une attaque à cryptogrammes choisis.

Exercice 9.7. *IND-CPA du chiffrement ElGamal*

a) Démontrer que si le sous-groupe Diffie-Hellman H_e, engendré par g et e, du groupe produit $G \times G$ est inextricable, alors le chiffrement ElGamal est IND-CPA.

b) Montrer que le chiffrement ElGamal n'est pas IND-CCA.

Chiffrement de Paillier

Ce chiffrement a été proposé par Pascal Paillier en 1999. Il a des propriétés homomorphiques qui le rendent intéressant pour certaines applications comme le vote électronique. Il repose sur l'inextricabilité du sous-groupe des classes de degré composite.

La définition de ce système de chiffrement est la suivante :

— Génération des clés : Soit $n = p \times q$ un entier égal au produit de deux nombres premiers distincts.
  — La clé publique est l'entier n.
  — La clé privée est constituée des facteurs (p, q).

— Chiffrement : Pour chiffrer un message m qui est un entier compris entre 1 et n :
  — Choisir un élément r aléatoire dans $\mathbb{Z}/n\mathbb{Z}^\star$.
  — Le cryptogramme est l'élément $E(m, r) = (1 + r)^m \times r^n$ dans $\mathbb{Z}/n^2\mathbb{Z}$.

Exercice 9.8. *IND-CPA du chiffrement de Paillier*

a) Comment reconstituer le message en clair m à partir du cryptogramme c et de la clé privée ?

b) Montrer que le chiffrement de Paillier a la propriété homomorphique suivante :

$$E(m_1, r_1) \times E(m_2, r_2) = E(m_1 + m_2, r_1 \times r_2).$$

Comment utiliser cette propriété pour réaliser un vote électronique ?

c) Le chiffrement de Paillier est-il IND-CCA ?

d) Soit $H = \{r^n \in \mathbb{Z}/n^2\mathbb{Z} \mid r \in \mathbb{Z}/n\mathbb{Z}\}$.
  — Montrer que H est un sous-groupe de $\mathbb{Z}/n^2\mathbb{Z}$.
  — Montrer que si le sous-groupe H est inextricable, alors le chiffrement de Paillier est IND-CPA.

5 Un chiffrement sûr à cryptogrammes choisis dans le modèle standard

Les cryptogrammes produits par les systèmes de chiffrement présentés dans le paragraphe précédent sont malléables. Cela signifie qu'un adversaire peut modifier un cryptogramme existant, puis le soumettre à un oracle de déchiffrement pour finalement récupérer des informations sur le message en clair, et gagner à l'un des jeux de la confidentialité.

Comme pour le chiffrement symétrique, pour rendre un système de chiffrement résistant aux attaques à cryptogrammes choisis, il faut que le déchiffrement puisse vérifier la conformité du cryptogramme et rejeter ceux qui semblent avoir été forgés par un adversaire. Un code d'authentification ne convient pas, car la clé du code doit être accessible pour le chiffrement et donc rester publique. La fonction qui est nécessaire est une sorte de signature inversée, avec une clé publique pour produire une signature, mais avec une vérification uniquement accessible au détenteur d'une clé privée. On parle alors de *vérificateur désigné*. Ce paragraphe présente un système de chiffrement qui a été conçu pour résister aux attaques par cryptogrammes choisis de type CCA2, tout en utilisant une fonction de hachage réaliste, dont on exige seulement qu'elle soit résistante aux collisions [1]. Ce système a été proposé par Ronald Cramer et Victor Shoup en 1998. Sa sécurité repose sur l'inextricabilité des sous-groupe Diffie-Hellman.

Preuve avec vérificateur désigné

Le principe d'un chiffrement CCA à clé publique a été proposé la première fois par Naor et Youg en 1991. Il consiste à chiffrer le message de deux façons différentes, avec une preuve qu'il s'agit du chiffrement du même message. Lorsqu'on adapte ce principe, par exemple au chiffrement ElGamal sur le groupe cyclique $(G, \times)$ d'ordre q premier, cela conduit à produire deux indices u et u' à partir du même aléa r, mais à partir de deux générateurs différents g et g' du groupe dans lequel on opère. Le cryptogramme a alors la forme suivante :

$$(u' = g^r,\ u' = g'^r,\ \gamma = e^r \times m,\ p),$$

1. Le chiffrement à clé publique appartenant par nature à Cryptomania, il n'est pas utile de faire une hypothèse plus faible de résistance au second antécédent

où p est une preuve que c'est bien le même aléa r a servi à produire les deux indices u et u', c'est-à-dire une preuve que $\log_g(u) = \log_{g'}(u')$. Cela revient à une preuve que le couple (u, u') appartient à l'ensemble $\mathcal{L} = \{(g^r, g'^r) \mid r \in \mathbb{Z}/q\mathbb{Z}\}$, qui est justement un sous-groupe Diffie-Hellman du groupe produit $G \times G$. La valeur de r est un témoin d'appartenance à $\mathcal{L}$, mais il ne doit pas être révélé, sans quoi le cryptogramme pourrait être déchiffré par tous grâce à la clé publique de chiffrement.

Le système de preuve fonctionne ainsi :

— Au chiffrement, une preuve p est calculée à l'aide du témoin r et d'une clé publique.

— Au déchiffrement, une valeur de vérification v, qui devra être égale à p, est calculée à l'aide du couple (u, u') et d'une clé privée.

Ce système de preuve empêche un adversaire de modifier des cryptogrammes. Bien-sûr, il pourra toujours produire des cryptogrammes valides en chiffrant un message avec la clé publique, mais l'oracle de déchiffrement lui rendra ce qu'il vient de chiffrer, ce qui ne lui sera d'aucune utilité pour répondre au défi qui lui est posé.

Le système de chiffrement

Les paramètres de ce système sont :

— Un groupe cyclique $(G, \times)$ d'ordre premier q. Le paramètre de sécurité est le nombre de symboles binaires de l'entier q.

— Deux générateurs g et g' de ce groupe.

— Une fonction de hachage $H : G^3 \to G$, résistante aux collisions.

La définition du système de chiffrement est :

— Génération des clés :

  — La clé privée est constituée de trois couples (x, x'), (y, y') et (z, z') de $\mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$. Le premier couple (x, x') est la clé privée de confidentialité et les deux autres, (y, y') et (z, z'), sont des clés privées de vérification de preuve.

  — La clé publique est constitué des trois éléments e, a et b suivants de G :

$$\begin{aligned} e &= g^x \times g'^{x'} \\ a &= g^y \times g'^{y'} \\ b &= g^z \times g'^{z'} \end{aligned}$$

  Le premier élément e est la clé publique de confidentialité et les deux autres, a et b, sont les clés publiques utilisées pour produire la preuve.

— Chiffrement : Pour chiffrer un message $m \in G$:

  — Choisir un élément r aléatoire dans $\mathbb{Z}/q\mathbb{Z}$.

  — Calculer :

$$\begin{aligned} c &= e^r \times m \\ u &= g^r \\ u' &= g'^r \\ h &= H(u, u', c) \\ p &= a^r \times b^{rh} \end{aligned}$$

  — Le cryptogramme est le quadruplet $\gamma = (u, u', c, p)$.

— Déchiffrement : Pour déchiffrer le cryptogramme (u, u', c, p) :

  — Calculer $h = H(u, u', c)$.

  — Utiliser la clé privée pour vérifier le cryptogramme : calculer $v = u^{y+hz} \times u'^{y'+hz'}$ et vérifier que $p = v$. Si ce n'est pas le cas, arrêter le déchiffrement et rejeter le cryptogramme.

  — Reconstituer le message en clair m en calculant le masque : $\mu = u^x \times u'^{x'}$, puis $m = c/\mu$.

Si le cryptogramme a été correctement formé, alors la vérification v est bien égale à p. En effet, $v = u^{y+hz} \times u'^{h'+hz'} = g^{r(y+hz)} \times g'^{r(y'+hz')} = (g^y \times g'^{y'})^r \times (g^z \times g'^{z'})^{hr} = a^r \times b^{hr} = p$. Par ailleurs, le message en clair est correctement reconstitué, car $\mu = u^x \times u'^{x'} = g^{rx} \times g'^{rx'} = (g^x \times g'^{x'})^r = e^r$.

Dans le cryptogramme (u, u', c, p), la composante c contient l'information du message en clair. L'information est masquée avec l'élément aléatoire e^r. Les données u et u' sont les deux indices, construits avec le même témoin aléatoire r, qui permettent au détenteur de la clé privée d'ôter le masque et de reconstituer ainsi le clair. La composante p est une information de preuve qui permet de vérifier que les deux indices appartiennent bien au sous-groupe Diffie-Hellman $\mathcal{L} = \{(g^r, g'^r) \mid r \in \mathbb{Z}/q\mathbb{Z}\}$. Cette preuve assure que l'émetteur du message a bien chiffré correctement le message et qu'il ne s'agit pas d'une tentative pour construire un faux cryptogramme et extorquer de l'information d'un oracle de déchiffrement.

Ce système de chiffrement a la propriété remarquable suivante :

Théorème 5.1. IND-CCA du chiffrement de Cramer-Shoup

Si le sous-groupe Diffie-Hellman est inextricable et si la fonction de hachage H est résistante aux collisions, alors le système de chiffrement de Cramer-Shoup est IND-CCA.

Preuve Supposons que la fonction H est résistante aux collisions et supposons que le chiffrement de Cramer-Shoup n'est pas IND-CCA, c'est-à-dire qu'il existe un adversaire A au jeu de l'indistinguabilité qui a accès à un oracle de déchiffrement et qui gagne avec un avantage non négligeable. Construisons un distingueur D qui reconnaît un élément du sous-groupe Diffie-Hellman auprès d'un oracle.

L'algorithme D ci-après joue au jeu du sous-groupe dans sa variante étoile contre un oracle et utilise l'adversaire A au jeu de l'indistinguablité comme sous-programme. L'oracle contre D choisit un symbole binaire $\widehat{b} \in_R \{0, 1\}$, puis génère un couple (u, u') aléatoire dans $\mathcal{L}$ si $\widehat{b} = 1$, et aléatoire dans $(G \times G) \setminus \mathcal{L}$ si $\widehat{b} = 0$.

Algorithme 5.2.

Entrée : Le couple (u, u') transmis par l'oracle.

1. D génère les clés privées x, x', y, y', z et z', calcule les clés publiques correspondantes qu'il transmet à A.
2. D reçoit de A deux messages m_0 et m_1. Il génère un symbole binaire aléatoire $b \in_R \{0, 1\}$.
3. D chiffre le message m_b en utilisant le couple (u, u') reçu de l'oracle et en générant c et p à l'aide des clés privées. Il transmet le cryptogramme $\gamma = (u, u', c, p)$ ainsi élaboré à l'adversaire A.
4. Lorsque A soumet une requête de déchiffrement $\gamma_i = (u_i, u'_i, c_i, p_i)$, l'algorithme D répond à A avec ses clés privées.
5. Après un temps polynomial, soit $b^\star$ l'estimation de b rendue par A.

Sortie : Si $b = b^\star$ alors renvoyer $\widetilde{b} = 1$ et si $b \neq b^\star$ alors renvoyer $\widetilde{b} = 0$.

L'algorithme D repose sur l'heuristique suivante : si l'adversaire A a trouvé la bonne valeur de b, c'est probablement que le cryptogramme γ_b qui lui a été transmis comme défi était correctement chiffré, c'est-à-dire $(u, u') \in \mathcal{L}$, et si A a perdu, c'est probablement que le cryptogramme n'était pas conforme, c'est-à-dire $(u, u') \notin \mathcal{L}$.

Il reste à évaluer l'avantage du distingueur D ainsi défini et à montrer que cet avantage est non négligeable. Par définition :

$$\mathrm{Av}(D) = \left| \Pr[\widetilde{b} = 1 \mid \widehat{b} = 1] - \Pr[\widetilde{b} = 1 \mid \widehat{b} = 0] \right|$$

Le premier terme est la probabilité que l'adversaire A gagne alors que le cryptogramme γ_b était correct. Il s'agit donc de la probabilité de succès de A.

Le second terme est la probabilité que A gagne alors que le cryptogramme γ_b n'était pas correctement constitué.

On a :

$$\mathrm{Av}(D) \geq \left|\left|\Pr[\widetilde{b} = 1 \mid \widehat{b} = 1] - \frac{1}{2}\right| - \left[\Pr[\widetilde{b} = 1 \mid \widehat{b} = 0] - \frac{1}{2}\right|\right|$$

Le premier terme est $\mathrm{Av}(A)/2$ qui est par hypothèse non négligeable. Il suffit de démontrer que le second terme est négligeable pour conclure. On s'intéresse donc maintenant au cas où $\widehat{b} = 0$, c'est-à-dire lorsque le défi $\gamma = (u, u', c, p)$ est tel que $(u, u') \notin \mathcal{L}$.

La fin de la preuve requiert deux lemmes. Le premier énonce que, si $\widehat{b} = 0$, c'est-à-dire si $(u, u') \notin \mathcal{L}$, alors les seules requêtes de déchiffrement soumise par A qui peuvent l'aider à conclure sont les requêtes $\gamma_i = (u_i, u'_i, c_i, p_i)$ pour lesquelles on a $(u_i, u'_i) \notin \mathcal{L}$.

Lemme 5.3.

Si dans le défi, on a $(u, u') \notin \mathcal{L}$ et si dans une requête de l'adversaire A, on a $(u_i, u'_i) \in \mathcal{L}$, alors la réponse de l'oracle de déchiffrement est indépendante de b.

Le second lemme énonce que la probabilité qu'une requête satisfait $(u_i, u'_i) \notin \mathcal{L}$ tout en passant la vérification, c'est-à-dire soit utile à A pour conclure, est négligeable.

Lemme 5.4.

Si $(u, u') \notin \mathcal{L}$, si la requête de déchiffrement numéro i satisfait $(u_i, u'_i) \notin \mathcal{L}$ et si la fonction de hachage H est résistante aux collisions, alors la probabilité que cette requête passe la vérification est négligeable.

En admettant provisoirement les résultats de ces deux lemmes, terminons la preuve du théorème 1.

D'après le lemme 3, les seules requêtes qui apportent une information à l'adversaire A sont celles pour lesquelles on a $(u_i, v_i) \notin \mathcal{L}$. D'après le lemme 4, la probabilité qu'une telle requête advienne a une valeur ε négligeable.

Soit m le nombre de requêtes de déchiffrement formulées par l'adversaire A. La probabilité qu'aucune de ces requêtes ne soit utile à A est $(1-\varepsilon)^m$ qui est supérieur à $1 - m\varepsilon$. La probabilité qu'au moins une requête de déchiffrement apprenne quelque chose à A est donc inférieure à $m\varepsilon$ qui est négligeable comme produit d'un facteur négligeable et d'un facteur majoré par un polynôme du paramètre de sécurité.

L'événement E : « *Il existe une requête de déchiffrement dont la réponse n'est pas indépendante de* b » a donc une probabilité négligeable η de survenir. Par conséquent :

$$\Pr[\widetilde{b} = 0 \mid \widehat{b} = 0] = \underbrace{\Pr[\widetilde{b} = 1 \mid \widehat{b} = 0 \text{ et } E]}_{= 1/2} \times \underbrace{P(E)}_{1-\eta} + \underbrace{\Pr[\widetilde{b} = 1 \mid \widehat{b} = 0 \text{ et } \overline{E}]}_{0 \leq \cdot \leq 1} \times \underbrace{P(\overline{E})}_{\eta} \tag{9.1}$$

Si l'événement E survient, le cryptogramme étant aléatoire, il a autant de probabilité d'être produit à partir de m_0 qu'à partir de m_1. La probabilité que A gagne dans ce cas vaut donc $1/2$.

On déduit de l'égalité 9.1 que :

$$\frac{1}{2} - \eta \leq \Pr[\widetilde{b} = 1 \mid \widehat{b} = 0] \leq \frac{1}{2} + \eta,$$

donc

$$\left|\Pr[\widetilde{b} = 1 \mid \widehat{b} = 0] - \frac{1}{2}\right| \leq \eta,$$

qui est négligeable et achève la preuve. □

Il reste maintenant à prouver les deux lemmes.

Preuve du lemme 3 Posons $g' = g^\omega$, où $\omega = \log_g(g')$. Par hypothèse, le couple des indices (u, u') du défi γ n'appartient pas à $\mathcal{L}$, donc $u = g^r$ et $u' = g'^{r'}$, avec $r \neq r'$. La valeur $\mu \in G$ qui masque le message m_b dans le défi est $\mu_b = g^{rx} \times g'^{r'x'} = g^{rx+\omega r'x'}$. Posons $\mu = g^\beta$. Le couple (x, x') de la clé privée de déchiffrement appartient donc à la droite d d'équation $rx + \omega r'x' = \beta$.

Posons $u_i = g^{r_i}$ et $u'_i = g'^{r_i} = g^{\omega r_i}$. La réponse de l'oracle déchiffrement informe l'adversaire que le masque μ_i qui servi à camoufler son message m_i est $\mu_i = g^{r_i x + \omega r_i x'} = g^{r_i(x+\omega x')}$. Posons $\mu_i = g^{\beta_i}$. Le couple (x, x') appartient donc à la droite d_i d'équation $x + \omega x' = \beta_i / r_i$. Les deux droites d et d_i ont des directions différentes, ce qui signifie que toutes les valeurs sont possibles pour β et donc pour le masque μ du défi. Ce masque est donc aléatoire. Le cryptogramme γ a autant de chance d'être celui de m_0 que celui de m_1. □

Preuve du lemme 4 Posons comme ci-dessus $g' = g^\omega$. Par hypothèse, $u = g^r$ et $u' = g^{\omega r'}$ avec $r \neq r'$ et si $\gamma_i = (u_i, u'_i, c_i, p_i)$ est le cryptogramme de la requête de chiffrement numéro i, on a $u_i = g^{r_i}$ et $u'_i = g'^{r'_i} = g^{\omega r'_i}$ avec $r_i \neq r'_i$. Posons $h = H(u, u', c)$ et $h_i = H(u_i, u'_i, c_i)$. Les clés publiques a et b, les preuves p et p_i sont reliées par les relations suivantes :

$$\begin{aligned} a &= g^{y+\omega y'} \\ b &= g^{z+\omega z'} \\ p &= g^{ry+\omega r'y'+hrz+\omega hr'z'} \\ p_i &= g^{r_i y+\omega r'_i y'+h_i r_i z+\omega h_i r'_i z'} \end{aligned}$$

L'application qui au quadruplet de clés privées (y, y', z, z') associe le quadruplet des logarithmes $(\log_g(a), \log_g(b), \log_g(p), \log_g(p_i))$ est donc linéaire et a pour matrice :

$$\begin{pmatrix} 1 & \omega & 0 & 0 \\ 0 & 0 & 1 & \omega \\ r & \omega r' & hr & \omega hr' \\ r_i & \omega r'_i & h_i r_i & \omega h_i r'_i \end{pmatrix}$$

Le déterminant de cette matrice vaut :

$$D = \omega^2 (r - r')(r_i - r'_i)(h - h_i)$$

Si $h \neq h_i$, alors cette application est injective et le quadruplet (a, b, p, p_i) est aléatoire, car en bijection avec le quadruplet aléatoire des clés privées. En particulier, p_i est aléatoire et indépendant de a, b et p. La probabilité que la preuve p_i passe la vérification ne dépasse pas $1/q$ qui est négligeable.

Posons E_i l'événement « *le cryptogramme γ_i est valide* », et F l'événement « $h = h_i$ ». Comme H est supposée être résistante aux collisions, la probabilité de l'événement F est négligeable. Par conséquent, la probabilité de E_i est :

$$\Pr(E_i) = \underbrace{\Pr(E_i \mid F)}_{\leq 1} \times \underbrace{P(F)}_{\text{négligeable}} + \underbrace{\Pr(E_i \mid \overline{F})}_{\text{négligeable}} \times \underbrace{\Pr(\overline{F})}_{\leq 1}.$$

La probabilité de E_i est donc négligeable. □

Exercice 9.9. Le système de chiffrement de Cramer-Shoup reste-t-il sûr :
a) si la preuve p vaut seulement $p = a^r$, sans fonction de hachage H, ni clé privée (z, z'), ni clé publique b ?
b) si la preuve p est constituée des deux termes (a^r, b^{rh}) au lieu de leur produit ?

Bibliographie

[1] Mihir Bellare et Philip Rogaway : Introduction to modern cryptography. `https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf`, 2005. [en ligne le 26 août 2019].

[2] David Bleichenbacher : Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs#1. *Proceeding of CRYPTO'98, LNCS 1462*, pages 1–12, 1998. `http://archiv.infsec.ethz.ch/education/fs08/secsem/Bleichenbacher98.pdf`.

[3] Whitfied Diffie et Martin Hellman : New directions in cryptography. *IEEE Transaction on Information Theory, Vol 22, no 6*, pages 644–654, 1976.

[4] Oded GOldreich : The foundations of cryptography. `http://www.wisdom.weizmann.ac.il/~oded/frag.html`. [en ligne le 26 août 2019].

[5] Shafi Goldwasser et Mihir Bellare : Lecture notes on cryptography at the mit. `http://www-cse.ucsd.edu/users/mihir/papers/gb.html`, 2008. [en ligne le 26 août 2019].

[6] Martin Hellman : A cryptanalytic time-memory trade-off. *IEEE Transaction on Information Theory*, 26(4):401–406, 1980.

[7] Russel Impagliazzo : A personal view of average-case complexity. *Proceeding of the 10th IEEE Conference on Structure in Complexity Theory*, pages 134–147, 1995.

[8] Jonathan Katz et Yehuda Lindell : *Introduction to Modern Cryptography.* Chapnan & Hall/CRC, 2014.

[9] Leslie Lamport : Constructing digital signatures from a one-way function. *Technical Report SRI-CSL-98, International Computer Science Laboratory.*

[10] Ronals Rivest, Adi Shamir et Leonard Adleman : A method for obtaining digital signatures and public-key cryptosystems. *Communications or the ACM,* vol. *21, n° 2*, pages 120–126, 1978.

Index