UNIVERSITÉ PARIS 8
VINCENNES-SAINT-DENIS

Mathématiques, Informatique, Technologies, Sciences de l'Information et de la Communication

Fondements théoriques de la cryptographie

Hieu Phan & Philippe Guillot

4 mars 2019

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 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 a peu de chance d'être d'une grande robustesse. Afin de profiter des experts cryptanalystes et autres 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. 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 de 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 énorme qu'il n'est que rarement 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 Roland Rivest en 1988. Il a été considéré comme sûr pendant dix ans, jusqu'à

ce 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 Bleidhenbacher publie un article décrivant comment un adversaire peut déchiffrer un message de son choix par une attaque à cryptogrammes choisis.

Aujourd'hui, des garanties de sécurité sont exigées dans la durée. La boucle infinie des attaques et des correctifs n'est pas satisfaisante. L'approche moderne de la conception des algorithmes cryptographiques revendique une approche scientifique. Dans la préface de leur ouvrage *Introduction to Modern Cryptography* [1] publié en 2008, les auteurs Oded Goldreich, 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.*

La clarté des énoncés, l'hypothèse cryptographique bien définie, nécessite 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 et un paramètre secret, une clé, 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 également être clairement définis et assignés. 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 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. 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 ?

Et pourtant, les preuves de sécurité sont loin d'être inutiles. Elles affirment quand-même 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 sans limite, et on peut compter sur leur inventivité pour trouver les moyens de faire sauter les verrous de sécurité mis en place. La preuve met en avant le 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, c'est-à-dire, étant donnée une fonction $f : E \to F$, de répondre à la question : « Cette fonction est-elle à sens unique ? »

La définition formelle repose sur un jeu qui se joue à deux : le maître du jeu, appelé aussi *oracle*, met au défi un challenger de trouver un antécédent à une image donnée. L'objectif du challenger est de trouver un antécédent à une valeur proposée par l'oracle. Les règles sont précisées ci-après :

Jeu 1.1. jeu des fonctions à sens unique

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*.

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 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 des données de taille variable. Elles peuvent être concrètement considérées comme des fonctions à sens unique.
- 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, appelé *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 $\varepsilon = \max_{A \in \mathcal{A}_t}(P_{succes}(A))$. 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 le maître du jeu 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 Hellmann (1945 -) [2]. 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 $z_i = f^i(x_i)$.
   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 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 cette probabilité elle tende vers zéro, mais on exige encore davantage. Une fonction est négligeable si elle décroit 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 dans le 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 le maître du jeu dans le 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.

Dans la suite de cet exposé, lorsque le terme *fonction à sens unique* apparaîtra sans autre qualificatif, il s'agira toujours de *fonctions asymptotiquement à sens unique*.

La probabilité est relative au choix aléatoire par le maître du jeu 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 $\{1,\ldots,n\}$ vers l'ensemble $\{1,2,3\}$ tel que deux sommets adjacents n'ont pas la même image, c'est-à-dire :

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

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 une notion 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 avec un paramètre, appartenant à un ensemble d'indices 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 est inférieure ou égale à n. 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 le nombre de chiffres binaires de l'entier n.

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 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. 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 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.

Exemple : formalisation de la famille de fonctions à sens unique MUL. Formaliser une famille de fonction à sens unique pratique consiste à préciser les éléments de la définition 1 qui lui correspondent.

- L'ensemble d'indices I est l'ensemble $\mathbb{N}$ des entiers naturels. Un indice est égal au paramètre de sécurité. L'algorithme $\mathcal{S}_1$ est donc l'identité. Pour l'entrée $n \in \mathbb{N}$, il produit le résultat n.
- Pour tout indice n le domaine D_n est l'ensemble des couples d'entiers premiers (p, q) dont le produit comprend n chiffres binaires. L'algorithme $\mathcal{S}_2$ produit un tel couple. Par exemple, une chaine binaire aléatoire r de n symboles binaires est décomposée en la concaténation de deux chaînes plus courtes : $r = r_1 \mid r_2$. Les chaînes r_1 et r_2 binaires sont interprétées comme des entiers écrits en représentation binaire. Les entiers p et q sont entiers premiers qui suivent respectivement r_1 et r_2, comme résultat de la fonction `next_prime` qui renvoie l'entier premier qui suit son paramètre `r`. Cette fonction se programme en utilisant les tests de primalité classiques.
- Pour tout $n \in \mathbb{N}$, la fonction f_n est la multiplication, définie par $f_n(p, q) = p \times q$.

Exercice 1.7. Formaliser de même les fonctions RSA, EXP et SQR définies au début de ce paragraphe.

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 à sens unique asymptotique. Construisons 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 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 efficace. 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. Cet algorithme a donc un succès supérieur à A, ce qui montre que sous cette hypothèse, la famille (f_i) n'est pas à sens unique. □

Exercice 1.8. 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 1.9. 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.

Exercice 1.10. Soit $f : E \longrightarrow E$ une fonction à sens unique.

1. La fonction $f \circ f$ est-elle à sens unique ?
2. La fonction $g : x \longmapsto \big(f(x), f \circ f(x)\big)$ est-elle à sens unique ?
3. On suppose maintenant que f est bijective. La fonction $f \circ f$ est-elle à sens unique ?

6 Prédicat difficile

Le moindre exigible d'une fonction à sens unique est que la valeur ne révèle pas une certaine information sur l'entrée. Ce qu'on appelle un prédicat difficile (*hard core predicate*) est précisément une information sur l'entrée 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 pseudo-aléatoires, 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 :

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

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 fonction booléenne de l'entrée.

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. jeu du prédicat difficile

1. L'oracle choisit une valeur x aléatoire dans E avec la probabilité uniforme.
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 $b^\star = p(x)$, dans le cas contraire, il a perdu.

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

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

Exercice 1.13. Montrer que si l'on dispose d'un oracle qui, pour tout $x \in \mathbb{Z}/n\mathbb{Z}$ fournit le chiffre de parité de x à partir de la valeur de $y = \mathrm{RSA}(x)$, alors pour tout $x \in \mathbb{Z}/n\mathbb{Z}$, on peut retrouver efficacement x à partir de la valeur $y = \mathrm{RSA}(x)$.

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 x fournit le chiffre de poids fort de x à partir de la valeur $y = \mathrm{EXP}(x)$, alors pour tout $x \in \mathbb{Z}/p\mathbb{Z}$, on peut reconstituer entièrement la valeur de x à partir de $y = \mathrm{EXP}(x)$.

Les exercices 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 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 plus facile dans un sens donnée par cette proposition :

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. D'où :

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

L'avantage de B est donc 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.

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. Si ce n'était pas le cas, il existerait un algorithme polynomial A qui sait inverser la fonction g avec une probabilité de succès non négligeable. Utilisons cet algorithme pour construire un algorithme B qui inverse la fonction f.

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

Entrée : y fourni par l'oracle en calculant $y = f(x)$ 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 :

$$\mathrm{Pr}_{\mathrm{succes}}(B) \geq \mathrm{Pr}_{\mathrm{succes}}(A)$$

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 résoudre un système d'équations linéaires et de retrouver la valeur du vecteur x.

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 $1/2 + 1/(2n)$. 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 le système. Or, la probabilité de recevoir de l'algorithme A toutes les composantes exactes survient avec une probabilité égale à :

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

Cette quantité converge vers $e/2^n$ quand n tend vers $+\infty$. La probabilité de succès de l'algorithme B est négligeable et cela ne permet pas de prouver le résultat.

Il faut donc un algorithme plus subtil qui supporte de recevoir de l'algorithme A des résultats inexacts, pourvu que l'avantage de A reste non négligeable, et qui conserve une probabilité de non négligeable succès ainsi qu'une complexité polynomiale.

Considérons l'algorithme suivant B, destiné à inverser la fonction f. Il utilise l'adversaire A comme sous-programme.

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^{e} 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 :

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

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

Il faut montrer trois choses : trouver les paramètres de l'algorithme B pour que sa complexité reste polynomiale, que l'algorithme B inverse bien la fonction f et que sa probabilité de succès est non négligeable. Pour cela, les tâches à effectuer sont les suivantes :

1. Évaluer l'avantage de A. Ici, l'algorithme A est appelé avec une entrée contrainte qui n'est pas un couple (y, r) où $y = f(x)$ avec x aléatoire et r est aléatoire. 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, r) contrainte

Soit $\mu(n)$ la probabilité de succès de l'algorithme A sur une entrée (y, r) où $y = f(x)$ avec x et r aléatoires dans $\{0,1\}^n$. Pour une telle valeur de y, 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 la probabilité de succès de l'algorithme B converge vers $1/e$ lorsque n tend vers $+\infty$.

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 4, la probabilité $\Pr(S \mid C \wedge E_y)$ tend vers $1/e$.

— 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 5, la probabilité $\Pr(E_y)$ est supérieure à $\mu(n)/2$.

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 la survenue ou non de l'événement E_y. 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, $\Pr(x_i \neq x_{i,I}^\star) = \Pr(X_{i,I} = 1) < 1/2 - \mu(n)/2$.

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 :

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

Donc

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

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 :

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

Par conséquent :

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

L'inégalité de Bienaymé-Tchébychev permet d'établir la majoration suivante :

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

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 :

$$\Pr(x_i^\star \neq x_i) \leq \frac{1}{2^\ell \mu(n)^2}.$$

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

$$\Pr(x_i^\star \neq x_i) < \frac{1}{n}.$$

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 :

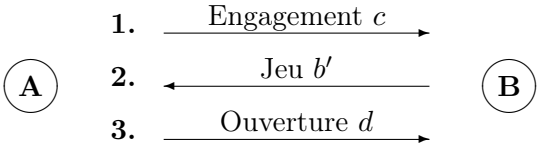
$$\Pr(x^\star = x) \geq \left(1 - \frac{1}{n}\right)^n.$$

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

8 Application : la 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** divulge 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 : $\text{OUV}(c, d) = b$.

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 $\text{OUV}(c, d) \neq \text{OUV}(c, d')$. 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 schéma de mise en gage pratique à l'aide d'une fonction à sens unique 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 pseudo-aléatoire sont :

— 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 pseudo-aléatoire*. Intuitivement, un générateur pseudo-aléatoire agit comme un amplificateur d'aléa.

Mais la théorie de l'information nous apprend que l'incertitude d'une séquence aléatoire ne peut pas résulter d'un calcul. L'entropie d'une séquence ne peut en aucun cas être supérieure à celle 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 pseudo-aléatoires.

1 Indistinguabilité des variables aléatoires

La qualité d'un générateur pseudo-aléatoire (GPA) se mesure à la difficulté de faire la différence entre l'observation issue de celui-ci d'une source issue 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 réalisation choisies parmi deux variables aléatoires, et un adversaire, appelé it distingueur, chargé de reconnaître de quelle variable sont issues les réalisations qui lui sont présentées.

Jeu 1.1. Jeu 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 $\Pr(b = b^\star)$.

Le jeu de l'aléa fait partie d'une famille de jeux dans lequel un oracle choisit un monde défini par le choix aléatoire d'un symbole binaire b dans l'ensemble $\{0,1\}$ avec la probabilité uniforme, et connu du seul oracle. Dans le monde 0, l'oracle fournit des réalisations de la variable X tandis que dans le monde 1, il fournit des réalisations de Y. Un distingueur est de manière générale un algorithme chargé de trouver dans quel monde l'oracle opère. Il renvoie une valeur $b^\star$ qui indique le monde dans lequel il crois que l'opacle opère.

La proposition suivante donne une expression générale de l'avantage d'un distingueur lorsque le monde choisis par l'oracle suit une loi équidistribuée, chaque monde ayant une probabilité égale à $1/2$ d'être choisi.

Proposition 1.2. Avantage d'un distingueur

Lorsque le choix du monde b est aléatoire et équidistribué, l'avantage d'un distingueur D qui renvoie l'estimation $b^\star$ a pour expression :

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

Preuve La probabilité de succès du distingueur D est $\mathrm{P}_{\text{succes}}(D) = \Pr(b = b^\star)$. Cette expression se décompose 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 $1 = \Pr(b^\star = 0 \mid b = 1) + \Pr(b^\star = 1 \mid b = 1)$, il vient :

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

□

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 $\mathrm{Av}_{X,Y}(D)$. Notons $\mathcal{D}$ l'ensemble de tous les distingueur.

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'avantage sur tous les distingueurs chargés de distinguer entre X et Y :

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

L'appellation *distance* pour cette quantité n'a rien de fortuit. Elle est justifiée par la proposition suivante :

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

Si les deux variables aléatoires X et Y sont de loi différentes, alors les ensembles A et C 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 $x \in A$, renvoyer $b^\star = 0$ qui signifie que x est probablement une réalisation de X.
2. si $x \in C$, renvoyer $b^\star = 1$ qui signifie que x est probablement une réalisation de Y.
3. sinon, c'est-à-dire si $x \in B$ renvoyer une valeur $b^\star$ aléatoire.

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

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

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

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

Lorsque $x \in B$ il est autant probable qu'il soit une réalisation de A ou de B, les termes (1) et (2) s'annulent donc. Par définition de A et B la différence des autres termes est strictement positive. L'avantage de l'algorithme A est donc non nul.

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

Inégalité triangulaire : Soit D un distingueur de $\mathcal{D}$. On note U la variable aléatoire dont il reçoit les réalisations lors du jeu de l'aléa. 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 D. L'expression de l'avantage donné par la relation 2.1 de la proposition 2 s'écrit :

$$\mathrm{Av}_{XZ}(D) = |\Pr(D = 0 \mid U = X) - \Pr(D = 0 \mid U = Z)|.$$

De l'inégalité triangulaire pour la valeur absolue, résulte une inégalité triangulaire sur $\mathrm{Av}_{XZ}(D)$:

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

De cette inégalité pour tout distingueur D, on en déduit l'inégalité triangulaire pour la distance calculatoire. Soit ε un réel strictement positif quelconque. Il existe un distingueur D_ε tel que la distance calculatoire $\Delta(X, Y)$ satisfait :

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

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 X et Y, on note XY la variable aléatoire dont les réalisation 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 :

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

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 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 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 choix 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 $k-1$ premiers termes sont des réalisations de X_n, le k^{e} terme est l'entrée u et les derniers termes 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.
4. Renvoyer la réponse $b^\star$ rendue par D.

L'heuristique qui guide cet algorithme est que 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 une réalisation 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 :

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

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 :

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

Comme le deuxième terme est non négligeable par hypothèse, il en est de même pour le premier, 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 pseudo-aléatoire

Un générateur pseudo-aléatoire 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 de termes. La valeur produite par un générateur pseudo-aléatoire est une réalisation de variable aléatoire $g(S)$, 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 pseudo-aléatoire

Un générateur pseudo-aléatoire (GPA) est une famille de fonctions $(g_n)_{n\in\mathbb{N}}$, 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 pseudo-aléatoire 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 n, qui est strictement inférieure 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 pseudo-aléatoire, 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 pseudo-aléatoire* un élément particulier d'un 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 fonction $g : \{0,1\}^n \to \{0,1\}^m$, avec $m > n$ est un générateur pseudo-aléatoire. Démontrer que la fonction :

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

est également un générateur pseudo-aléatoire.

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

Exercice 2.3. Soit $g : \{0,1\}^n \to \{0,1\}^{2n}$ un générateur pseudo-aléatoire d'expansion double. Démontrer que la fonction g est à sens unique.

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.

Un distingueur de générateur pseudo-aléatoire doit reconnaître un pseudo-aléa, qui appartient à l'image $\mathrm{Im}(g)$ de la fonction g, d'un véritable aléa qui peut appartenir à tout l'espace $\{0,1\}^{\ell(n)}$. Si l'entrée y qui lui est fournie appartient à l'image $\mathrm{Im}(g)$, il y a deux possibilités dans le jeu de l'aléa :

L'oracle a choisi $b = 0$ et nécessairement $y \in \mathrm{Im}(g)$. Avec les notations du jeu de l'aléa, on a :

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

L'oracle a choisi $b = 1$ et on peut avoir $y \in \mathrm{Im}(g)$ de manière fortuite.

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

Le distingueur n'a accès qu'à la réalisation y et n'a aucune connaissance de la valeur b choisie par l'oracle. Le succès du distingueur ne peut reposer que sur l'appartenance ou non de son entrée y à l'image de la fonction g. Un distingueur peut reconnaître $y \in \mathrm{Im}(g)$ comme pseudo-aléatoire alors qu'il est le réslutat d'un tirage véritablement aléatoire. Pour cette raison, son avantage ne peut pas atteindre la valeur 1. En effet, $\mathrm{Av}(D)$ est la valeur absolue de la différence des premiers membres des équations 2.3 et 2.4, soit :

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

Exercice 2.4. Soit g un générateur pseudo-aléatoire 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 $1 - 1/2^k$.

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

Soient g et h deux générateurs pseudo-aléatoires. Démontrer que la fonction

$$\begin{array}{rcl} \{0,1\}^\star & \to & \{0,1\}^\star \\ (x,y) & \mapsto & \big(g(x),h(y)\big) \end{array},$$

est un générateur pseudo-aléatoire.
Indication : utiliser la méthode hybride.

Exercice 2.6. *Composition de générateurs pseudo-aléatoires*

Soit :

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

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

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

est un générateur d'aléa d'expansion $3n$.

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

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

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

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

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

4 Constructions

L'exercice 3 ci-dessus montre qu'un générateur est aussi une fonction à sens unique, et donc que l'existence d'un générateur pseudo-aléatoire 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 pseudo-aléatoire 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 pseudo-aléatoire 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 pseudo-aléatoire $\{0,1\}^n \to \{0,1\}^{n+1}$ d'expansion un symbole binaire. Cela constitue la première étape d'une construction plus générale utilisables en pratique.

Théorème 4.1. générateur pseudo-aléatoire 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 :

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

est un générateur pseudo-aléatoire d'un symbole binaire d'expansion.

Aujourd'hui, la fonction RSA et son chiffre de parité permettent de construire un tel générateur pseudo-aléatoire. 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 pseudo-aléatoire. 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, construisons un algorithme A, 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 $b^\star = 1 - r$.

L'heuristique qui guide cet algorithme est que si le couple (y,r) est reconnu comme pseudo-aléa toire, c'est que r est la valeur du prédicat $p(x)$.

La fonction f étant bijective, l'entrée y de l'algorithme A suit la loi uniforme. Comme x et r sont aléatoires, l'entrée de D correspond à un choix $b = 1$ pour l'oracle du jeu de l'aléa. La réponse $d^\star$ de D suit donc une loi de probabilité définie par $\Pr(d^\star = 0 \mid b = 1)$. Décomposons cette probabilité selon que $p(x)$ est ou non égal à r :

$$\Pr(d^\star = 0 \mid b = 1) = \Pr(d^\star = 0 \mid p(x) = r) \times \Pr(p(x) = r) + \Pr(d^\star = 0 \mid p(x) \neq r) \times \Pr(p(x) \neq r)$$

Comme r est aléatoire, $\Pr(p(x) = r) = \Pr(p(x) \neq r) = 1/2$. Notons pour simplifier $\alpha = \Pr(d^\star = 0 \mid p(x) = r)$ et $\beta = \Pr(d^\star = 0 \mid p(x) \neq r)$. On a :

$$\Pr(d^\star = 0 \mid b = 1) = \frac{1}{2}(\alpha + \beta).$$

Notons également que si le choix de l'oracle au jeu de l'aléa est $b = 0$, alors $p(x) = r$, ce qui implique que $\Pr(d^\star = 0 \mid b = 0) = \Pr(d^\star = 0 \mid p(x) = r)$. L'avantage du distingueur D a donc pour expression :

$$\begin{aligned} \mathrm{Av}(D) &= |\Pr(d^\star = 0 \mid b = 0) - \Pr(d^\star = 0 \mid b = 1)| \\ &= \left|\alpha - \frac{1}{2}(\alpha + \beta)\right| \\ &= \frac{1}{2}|\alpha - \beta| \end{aligned}$$

Évaluons maintenant la probabilité de succès de l'algorithme A. Cette probabilité est par définition $\Pr(b^\star = p(x))$. Décomposons selon la valeur de $p(x)$:

$$\Pr(b^\star = p(x)) = \Pr(b^\star = p(x) \mid \Pr(p(x) = r) \times \Pr(p(x) = r) + \Pr(b^\star = p(x) \mid \Pr(p(x) \neq r) \times \Pr(p(x) \neq r)$$

Par la définition de l'algorithme, l'événement $b^\star = r$ correspond à $d^\star = 0$ et l'événement $b^\star \neq r$ à $d^\star = 1$, donc :

$$\begin{aligned} \Pr(b^\star = p(x)) &= \frac{1}{2}\Big(\underbrace{\Pr(d^\star = 0 \mid \Pr(p(x) = r)}_{= \alpha} + \underbrace{\Pr(d^\star = 1 \mid \Pr(p(x) \neq r)}_{= 1 - \beta}\Big) \\ &= \frac{1}{2} + \frac{\alpha - \beta}{2} \end{aligned}$$

On en déduit que l'avantage de l'algorithme A vaut $\alpha - \beta$ qui est le double de l'avantage du distingueur D. Si $\mathrm{Av}(D)$ est non négligeable, il en est de même pour $\mathrm{Av}(A)$. □

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 pseudo-aléatoire.

4.2 Générateur pseudo-aléatoire d'expansion polynomiale

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

Théorème 4.3. Générateur pseudo-aléatoire 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 pseudo-aléatoire 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 pseudo-aléatoire.

Preuve Pour prouver ce théorème, on utilise la technique hybride déjà mise en œuvre pour la preuve du théorème 2 page 28. 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 K_ℓ est H_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 pseudo-aléatoire.

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 pseudo-aléatoire g.

Algorithme 4.4. A **contre le générateur pseudo-aléatoire** 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.
5. 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, et si l'un est non négligeable, alors l'autre ne l'est pas non plus. $\mathrm{Av}(D)$ étant supposé non négligeable, $\mathrm{Av}(A)$ est non négligeable, ce qui contredit l'hypothèse selon laquelle g est un générateur pseudo-aléatoire. □

5 Application au chiffrement

Comme affirmé dans l'introduction de ce chapitre, un générateur pseudo-aléatoire permet le chiffrement d'un message long avec une clé courte. Plus précisément, un générateur pseudo-aléatoire $g : \{0,1\}^n \to \{0,1\}^\ell$ 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 du message. Il évite la transmission préalable d'une quantité d'aléa aussi importante que les messages à chiffrer ultérieurement. L'intérêt pratique est considérable.

Comme pour le masque jetable, on ne peut chiffrer qu'un seul message avec une clé donnée. pour chiffrer un autre message, il faut une autre clé. Ce mécanisme est finalement ce qu'on pourrait appeler un *système de chiffrement avec clé jetable.*

Les fonctions pseudo-aléatoires présentées au chapitre suivant permettront de résoudre ce problème et 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
Famille de fonctions pseudo-aléatoires

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 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 $|F|^{|E|}$.

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 de fonctions pseudo-aléatoires, par exemple à l'aide d'une clé secrète, revient pratiquement 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 de fonctions pseudo-aléatoires 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 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 Indistingabilité 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 indistinguable 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. Jeu de la 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.

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 :

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

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

Exercice 3.1. Soit 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 toujour 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.

3 Famille de fonctions calculatoirement indistinguables

De façon informelle, deux familles de fonctions seront dites calculatoirement indistinguable, 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 indistinguable si tout adversaire polynomial pour le jeu de la reconnaissance a un avantage négligeable en 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 de fonctions pseudo-aléatoires. Soit $\mathcal{H}$ la famille de toutes les fonctions $\{0,1\}^\star \to \{0,1\}^\star$.

Définition 3.2. Famille de fonctions pseudo-aléatoires

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.

En d'autres termes, cela signifie qu'il n'est possible à un adversaire de distinguer entre un élément aléatoire de $\mathcal{F}$ et une fonction totalement aléatoire sur le même domaine, qu'avec un avantage négligeable.

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

On peut définir la famille de fonctions pseudo-aléatoire selon un double paramétrage : d'une part le paramètre de sécurité n, et d'autre part, la valeur de n étant fixée, pour un indice k appartenant à un ensemble d'indice. Ce dernier indice k est à comprendre comme une clé

secrète. 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 de fonctions pseudo-aléatoires alors il existe un générateur pseudo-aléatoire*

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

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

est un générateur pseudo-aléatoire d'expansion double.

4 Famille de fonctions à sens unique

Une famille de fonctions à sens unique 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 de fonctions à sens unique

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 de fonctions à sens unique* 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.

Cela ne signifie pas que les éléments f_k de la famille sont des fonctions à sens unique. Connaissant la clé k, il n'y a aucune raison que ces fonctions soient difficiles à inverser. La définition stipule seulement qu'il n'est pas possible de trouver un antécédent d'un élément y sans connaissance de la clé k et lorsqu'on ne dispose que d'un oracle pour déterminer les valeurs $f_k(x)$.

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

5 Construction de Goldreich, Goldwasser et Micali

L'exercice 4 ci-dessus montre que s'il existe une famille de fonctions pseudo-aléatoires, alors cela permet de construire un générateur pseudo-aléatoire d'expansion double. L'objet de cette section est de montrer la réciproque, c'est-à-dire de proposer la construction d'une famille de fonctions pseudo-aléatoires à partir d'un générateur pseudo-aléatoire 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, Shafrira Goldwasser et Silvio Micali.

On suppose donc que l'on dispose d'un générateur pseudo-aléatoire 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 premiers et les n derniers symboles binaires de la valeur du générateur pseudo-aléatoire 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

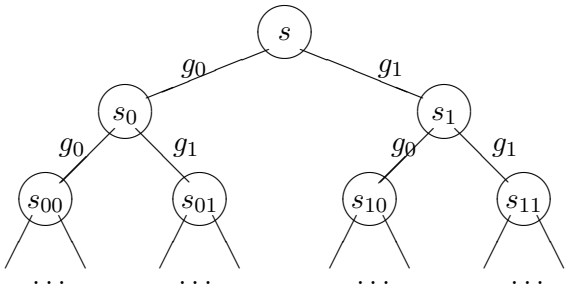
Si k est borné par un polynôme en n, et si g est un générateur pseudo-aléatoire, alors la famille $\mathcal{H}$ définie ci-dessus est une famille pseudo-aléatoires de fonctions.

La preuve de cette construction utilise la technique hybride.

Représentons cette construction par un arbre. La racine de l'arbre est le germe du générateur pseudo-aléatoire, 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 $\mathcal{H}_i$ de façons 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 $i \in \{0,\dots k\}$ et pour un mot binaire $u \in \{0,1\}^k$, décomposons $u = v \mid w$, 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 $u = v \mid w$ 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 $s_{0\dots0}\cdots s_{1\dots1}$.

La famille hybride $\mathcal{H}_i$ comprend une partie aléatoire représentée par l'indice $\sigma = (s_v)_{v\in\{0,1\}^i}$, et une partie pseudo-aléatoire. 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 en fait la famille $\mathcal{H}$, alors que la famille $\mathcal{H}_k$ est la famille de toutes les fonctions $\{0,1\}^k \to \{0,1\}^n$.

Montrons le lemme suivant qui est le cœur de la preuve de la proposition 1 :

Lemme 5.2.

Soit i compris entre 0 et $k-1$ Soit D un disgingueur 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 $i+1^{\text{e}}$ 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 $\alpha = a_0 \mid a_1$ 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 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 32 sur la distance calculatoire de la concatétation 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 calculatiores 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 $\{0,1\}^n$, on les définit sur $\{0,1\}^{\leq n}$. C'est-a-dire que $f(x)$ est défini pour des tailles de x variables de 1 jusqu'à n symboles binaires. Cela définit-il encore une famille de fonctions pseudo-aléatoire ?

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 $\left(f_k\right)_{k\in\{0,1\}^n}$ de fonctions booléennes $f_k : \{0,1\}^{p(n)} \longrightarrow \{0,1\}$, construire une famille de fonctions pseudo-aléatoire $\left(g_s\right)_{s\in S}$ de fonctions vectorielles $g_s : \{0,1\}^{p(n)} \to \{0,1\}^{\ell}$.

4 Chiffrement

Le chiffrement est le service traditionnel 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 message. 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 paramètres et des mécanismes qui permettent de chiffrer les messages afin d'empêcher à tout adversaire d'accéder à la moindre information 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 qui définit la taille des clés utilisées.

Définition 1.1. Système de confidentialité

Pour une valeur donnée d'un 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.
  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 de fonctionnement suivante :

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

La propriété de fonctionnement indique seulement 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 en 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

La sécurité sémantique est la version calculatoire de la sécurité parfaite. Cette dernière énonce que le cryptogramme n'apporte aucune information sur le clair. La version calculatoire énonce que le cryptogramme ne permet en pratique pas de calculer la moindre information sur le message en clair. En d'autres termes, tout ce qui est calculable sur le clair peut se calculer sans le cryptogramme. Si par contre le cryptogramme permet de calculer une certaine valeur binaire sur le 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 maître du jeu 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. Le maître du jeu choisit un message $m \in \mathcal{M}$ selon la loi de probabilités sur $\mathcal{M}$. Ce message est tenu secret.
2. Le maître du jeu choisit un couple de clés (e,d) aléatoire dans $\mathcal{K}$.
3. Le maître du jeu présente à l'adversaire le cryptogramme $\gamma = E_d(m)$.
4. L'adversaire renvoie au maître du jeu une estimation $b^\star$ de la valeur de $b(m)$.

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

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

$$\mathrm{Av}_{SEM}(A) = |\Pr(b^\star = b(m) - \Pr(b^\star = b(m')| \times \frac{1}{\Pr(b^\star \neq b(m')},$$

où m est le message choisit par le maître au cours du jeu, et m' est un message aléatoire dans $\mathcal{M}$. Ainsi, si l'adversaire répond au hasard, son avantage est nul.

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, étant donnée la distribution de la valeur de l'information sur les messages aléatoires. 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 équilibré si la valeur :

$$|\Pr(b(m) = 0 - \Pr(b(m) = 1)|$$

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 aléatoire indépendants de $\mathcal{M}$. Montrer que si la fonction $b \to \mathcal{M} \to \{0,1\}$ est quasi équilibrée, alors la quantité $\left|1 - 2\Pr\big(b(m_0) = b(m_1)\big)\right|$ 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 qu'avec un avantage négligeable.

Exercice 4.2. Le chiffrement RSA défini par $\gamma = m^e \bmod n$ est-il sémantiquement sûr ?

3 Indistingabilité

La sécurité sémantique est une notion assez délicate à établir, car elle soit 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'indistingabilité.

Cette notion repose sur un jeu qui oppose le maître du jeu à un adversaire. Les deux joueurs disposent de la connaissance d'un système de confidentialité.

Jeu 3.1. de l'indistinguabilité

1. Le maître du jeu 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 donne au maître du jeu.
3. Le maître du jeu choisit une valeur binaire aléatoire $b \in_{\mathrm{R}} \{0,1\}$ et chiffre le message m_b et renvoie à l'adversaire le cryptogramme correspondant.
4. L'adversaire dit de quel message m_0 ou m_1 le cryptogramme est issu, et pour cela propose une estimation $b^\star$ 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 :

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

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 le maître du jeu. On supposera pour simplifier que tous les messages et tous les cryptogrammes ont la même taille.

Si l'adversaire ne peut savoir quel message a été chiffré qu'avec un avantage négligeable, le schéma de chiffrement est dit indistinguable, ou encore qu'il a la propriété d'indistingabilité.

Définition 3.2. Système de confidentialité indistinguable

On dit qu'un système de confidentialité a la propriété d'indistingabilité si l'avantage de tout adversaire polynomial au jeu de l'indistingabilité 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 si 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 donné par le maître du jeu. Cette stratégie est toujours gagnante. Par conséquent :

Proposition 3.3.

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

De manière 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 schéma $\widetilde{\Sigma}$ est-il indistinguable ?
b) Est-il sémantiquement sûr ?

Exercice 4.4. *Indistinguabilité du mode ECB*

On dispose d'un schéma 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'indistingabilité ?

4 Équivalence de la sécurité sémantique et de l'indistingabilité

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

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

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

Preuve

Montrons tout d'abord qu'un système de confidentialité sémantiquement sûr a la propriété d'indistingabilité. Pour cela, montrons la contraposée de cette implication : $\neg IND \Rightarrow \neg SEM$.

Par hymothèse, il existe deux messages, m_0 et m_1 dont adversaire polynomial A sait distinguer le chiffré 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 du jeu de l'indistingabilité est aussi un adversaire qui peut jouer au jeu de la sécurité sémantique. Soit m le message choisi par le maître du jeu de la sécurité sémantique et soit $b^\star$ la réponse de l'adversaire à jeu. Soit aussi ε un élément aléatoire de $\{0,1\}$ choisi avec probabilité $1/2$. On a :

$$\begin{aligned} \mathrm{Av}_{SEM}(A) &= \Big|\Pr(b^\star = b(m) - \underbrace{\Pr(b^\star = b(m_\varepsilon)}_{=\frac{1}{2}}\Big| \times \underbrace{\frac{1}{\Pr(b^\star \neq b(m_\varepsilon)}}_{=2} \\ &= |2\Pr_{succes}(A) - 1| \\ &= \mathrm{Av}_{IND}(A). \end{aligned}$$

Il en résulte que l'adversaire A a le même avantage au jeu de la sécurité sémantique qu'au jeu de l'indistinguabilité. Comme ce dernier 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 indistingable, 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 s'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 négligeabme au jeu de l'indisinguabilité

Algorithme 4.2. Adversaire B au jeu de l'indistingabilité

- B choisit deux messages aléatoire m_0 et m_1 selon la loi de probabilité sur $\mathcal{M}$ et fournit ces deux messages au maître du jeu.
- B reçoit du maître du jeu le crypogramme c_b de m_b, pour une valeur de $b \in \{0,1\}$ aléatoire et inconnue.
- B transmet ce cryptogramme à l'adversaire A qui lui rénvoie une estimation $\widehat{b}$ de la fonction d'information sur m_b.
- Si $\widehat{b} = b(m_0)$ alors poser $b^\star = 0$,
- si $\widehat{b} = b(m_1)$ alors poser $b^\star = 1$
- et sinon donner à $b^\star$ une valeur aléatoire de $\{0,1\}$.
- Renvoyer la valeur de $b^\star$.

L'heuristique qui guide cet algorithme est que 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 :

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

L'événement $b^\star = 0$ survient si $b(m_0) = \widehat{b}$ ou avec probabilité $1/2$ si $b(m_0), b(m_1) \neq \widehat{b}$. Par conséquent :

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

De même,

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

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 b :

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

Par différence des équations 4.1 et 4.2, on a :

$$\mathrm{Av}(B) = \left|\Pr(\widehat{b} = b(m_0) \mid b = 0) - \Pr(\widehat{b} = b(m_0) \mid b = 0)\right|$$

Le premier terme est la probabilité de succès de l'adversaire A. Comme m_0 est aléatoire, le second terme est la probabilité que A réponde la valeur de la fonction d'information d'un message aléatoire indépendant de son entrée. Donc :

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

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

Rappelons que l'avantage des adversaires est une fonction d'un paramètre de sécurité n qui est sous-entendu dans toutes les écritures et que le caractère négigeable est une propriété asymptotique lorsque n tend vers $+\infty$.

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 choisie, 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.

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 lors du jeu de l'indistingabilité.

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.

Taxinomie des attaques Cette taxinomie des attaques relativement au jeu sécurité – indistingabilité ou sécurité sémentique – 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'indistingabilité lors d'une attaque à clairs choisis.
- Un système de confidentialité est dit SEM-CCA s'il est sémantiquement sûr lors d'une attaque à cryptogrammes choisis.

Exercice 4.5. *Sur le mode CBC*

a) Démontrer qu'un chiffrement CBC avec vecteur initial prédictible n'est pas IND-CPA.
b) démontrer qu'un chiffrement CBC avec vecteur initial aléatoire n'est pas IND-CCA.

Exercice 4.6. *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 $\{0,1\}^n \to \{0,1\}^m$. On considère le schéma de confidentialité suivant :

- L'ensemble des messages est $\mathcal{M} = \{0,1\}^n$.
- L'ensemble des cryptogrammes est l'ensemble des couples $\mathcal{C} = \{0,1\}^m \times \{0,1\}^m$.
- Le système est symétrique et l'ensemble des clés est l'ensemble $\mathcal{S}$ des clés 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 schéma de confidentialité n'est pas IND-CCA.
c) Démontrer que ce schéma de confidentialité est IND-CPA.

Exercice 4.7. *Un chiffrement à flot IND-CPA de messages de taille variable*

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

- L'ensemble des messages est l'ensemble des mots binaires de longueur arbitraire $\mathcal{M} = \{0,1\}^\star$.
- L'ensemble des cryptogrammes est l'ensemble des couples $\mathcal{C} = \mathbb{Z}/n\mathbb{Z} \times \{0,1\}^\star$.
- Le schéma est symétrique et son ensemble des clés est l'ensemble $\mathcal{S}$ des indices de la famille de fonctions.
- La fonction de chiffrement est $E_s :\ m = m_0 m_1 \cdots m_\ell \mapsto \big(r, m_0 \oplus f_s(r) \mid m_1 \oplus f_s(r+1) \mid \cdots \mid m_\ell \oplus f_s(r+\ell)\big)$

a) Décrire une fonction de déchiffrement.
b) Ce schéma est-il IND-CCA ?
c) Ce schéma est-il IND-CPA ?

5 Codes d'authentification

Les schémas 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 r 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 :

$$\begin{aligned} c = m \oplus f_s(r) &= c' = m' \oplus f_s(r) \\ c \oplus c' &= m \oplus m' \end{aligned}$$

Ainsi, un adversaire pourra requérir le déchiffrement d'un cryptogramme modifié pour extorquer des informations sur le message en clair et répondre ainsi avec certitude au défi lancé lors du jeu de l'indistingabilité.

Cette propriété de pouvoir manipuler les cryptogrammes pour en faire d'autres cryptogrammes valides s'appelle la *malléabilité*. Un chiffre malléable n'est jamais sûr contre une attaque à cryptogramme choisi. Pour qu'un chiffre soit sûr, il est nécessaire qu'il ne soit pas malléable. La non malléabilité signifie que si l'adversaire change la moindre parcelle d'un cryptogramme, il ne doit pas pouvoir obtenir de relation sur les messages en clair.

Une façon de procéder est d'ajoindre au cryptogramme une empreinte qui permet de s'assurer de l'authenticité du cryptogramme afin rendre impossible sa modification. Pour satisfaire cette exigence d'authenticité, l'empreinte aussi doit dépendre d'un paramètre secret. Si l'authenticité n'est pas assurée, le cryptogramme sera rejeté, ne sera pas déchiffré, et aucune information ne sera accessible sur le clair. La primitive qui assure cette fonction s'appelle un *code d'authentification* ou MAC (*Message Authentication Code*).

1 Définition

Un code d'authentification est une primitive cryptographique qui calcule une empreinte d'un message à l'aide d'une clé secrète. Il s'agit en quelque sorte d'une signature symétrique où la clé de signature est la même que la clé de vérification de cette signature. Ce type de mécanisme assure l'authentification du message dans la mesure où la production d'une empreinte vérifiable n'est possible que par le détenteur de la clé. 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'empreinge du document et non pas un vérificateur malhonête. 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 sécurite, 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 générer et pour vérifier l'empreinte, et si la fonction de génération d'empreinte est déterministe, on peut toujours supposer que la vérification est calculée selon la procédure suivante :

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

Cette procédure est appelée *fonction de vérification générique*. Elle ne fonctionne bien sûr pas si la fonction de génération de code n'est pas déterministe.

2 Sécurité

Lasécurité d'un code d'authentification de message se mesure à la difficulté de forger une empreinte valide sans connaître la clé secrète lors d'une attaque à messages choisis. L'objectif de l'adversaire est de construire un message et une empreinte valide qui passera l'algorithme de vérification, et ceci sans disposer de la clé secrète qui permet de produire ces empreintes. Pour cela, il peut interroger qui lui fournira des empreintes valides de messages de son choix. Bien sûr, le message fourni par l'adversaire à la fin du jeu doit être différent de tous les messages qu'il a adressés à l'oracle. La sécurité d'un code d'authentification se formalise par un jeu entre un oracle et un adversaire.

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 fourni $(m, c) \in \mathcal{M} \times \mathcal{C}$, où m est différent de tous les m_i qu'il a soumis.

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 des probabilités de succès de tous ses adversaires. En notant $\mathcal{A}$ l'ensemble de tous les adversaires au jeu de l'authentification, la sécurité du MAC est donnée par :

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

Définition 2.2. MAC sûr

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

Une famille de fonctions pseudo-aléatoires définit un code d'authentification de messages, comme le montre l'exercice qui quit.

Exercice 5.1. Soit $\mathcal{F} = (f_s)_{s \in S}$ une famille 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 médiocre. Il ne fonctionne que sur un ensemble de messages réduits. Le CBC-MAC permet de produire des empreintes de taille fixe pour des messages bien plus longs.

4 Chiffrement IND-CCA générique

La combinaison d'un schéma de chiffrement IND-CPA et d'un code d'authentification sûr conduit à un schéma de chiffrement IND-CCA, et ceci indépendamment du schéma de confidentialité choisi.

1. Considérons un schéma de chiffrement 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 schéma de chiffrement 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 construit un schéma de chiffrement $\Gamma^\star$ défini 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 de est l'ensemble $\mathcal{E}\times\mathcal{S}$ des couples (c,σ) constitués du cryptogramme pour Γ et d'une empreinte pour 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 $D'_{k,k'}$ vérifie tout d'abord que l'empreinte σ est bien celle du cryptogramme c, c'est-à-dire que $V_{k'}(c,\sigma)=1$. Si ce n'est pas le cas, elle renvoie un symbole $\perp$ pour signifier l'échec du déchiffrement. Dans le cas contraire, elle déchiffre le cryptogramme c avec la fonction de déchiffrement de Γ en calculant le message en clair $m=D_k(c)$ qu'elle renvoie.

L'adjonction d'une empreinte d'authentification de l'information chiffrée rend le cryptogramme non malléable et lui confère une sécurité contre une attaque à cryptogramme choisis comme l'énonce le théorème suivant :

Théorème 4.1. Chiffrement IND-CCA générique

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

Preuve Supposons que le schéma $\Gamma^\star$ n'est pas IND-CCA, c'est-à-dire qu'il existe un adversaire A, polynomial, contre le jeu IND-CCA et qui a un avantage non négligeable. Montrons alors que le schéma de chiffrement Γ n'est pas IND-CPA ou le code d'authentification Σ n'est pas sûr.

Pour cela, construisons un adversaire B contre le jeu IND-CPA qui utilise A comme sous-programme. L'algorithme B utilise également le code d'authentification Σ. Il génère une clé k' d'authentification. L'algorithme B fait appel à l'oracle de chiffrement pour le chiffrement des deux messages m_0 et m_1 transmis par A et également pour ses requêtes de chiffrement. Il reçoit les cryptogrammes auxquels il adjoint l'empreinte d'authentification. Il maintient également une liste $\mathcal{L}$ de couples clairs-cryptogrammes obtenus au cours de ces requêtes.

Pour les requêtes de déchiffrement, il vérifie les empreintes. 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 est dans sa liste $\mathcal{L}$ et 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 $\perp$.

L'algorithme B reçoit la réponse $b^\star$ de A qu'il renvoie lui-même comme valeur de retour.

Par définition, l'avantage de l'algorithme A est :

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

Soit E l'événement suivant : « *Il existe un cryptogramme valide parmi les requêtes de déchiffrement formulées par A.* »

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

$$\begin{aligned}
\mathrm{Av}(A) &= \left|2\Pr(b=b^\star\mid E)\times\Pr(E)+2\Pr(b=b^\star\mid\overline{E})\times\Pr(\overline{E})-1\right|\\
\mathrm{Av}(A) &= \left|2\Pr(b=b^\star\mid E)\times\Pr(E)+2\Pr(b=b^\star\mid\overline{E})\times\Pr(\overline{E})-P(E)-P(\overline{E})\right|\\
&\leq \left|\Pr(E)\Big(2\Pr(b=b^\star\mid E)-1\Big)\right|+\left|\Pr(\overline{E})\Big(2\Pr(b=b^\star\mid\overline{E})-1\Big)\right|
\end{aligned}$$

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 comme un véritable oracle de déchiffrement pour le jeu INC-CCA. Dans ce cas, le succès de A correspond exactement au succès de B. Cela signifie que la quantité :

$$\left|2\Pr(b = b^\star \mid \overline{E}) - 1\right|$$

est exactement l'avantage de B dans le jeu IND-CPA, d'où la majoration de l'avantage de A, obtenue en majorant $\Pr(b = b^\star \mid R)$ et $\Pr(\overline{E})$ par 1 :

$$\mathrm{Av}(A) \leq P(E) + \mathrm{Av}(B).$$

Soit p la probabilité qu'un cryptogramme produit par A soit valide, et soit K le nombre de requêtes de déchiffrement qu'il a formulées. On a :

$$P(\overline{E}) = (1-p)^K \approx 1 - Kp$$

Donc :

$$P(E) \approx Kp$$

On en déduit :

$$\mathrm{Av}(A) \leq Kp + \mathrm{Av}(B)$$

L'adversaire A étant supposé polynomial, son avantage étant supposé non négligeable, cela implique, ou que p est non négligeable, ou que l'avantage de B est non négligeable, ou les deux, et c'est ce qu'il fallait démontrer. □

6 Chiffrement par bloc

7
Signature numérique

8

Chiffrement à clé publique

9
Corrigé des exercices

Exercice 1, page 6. Non, car l'algorithme qui à y nul renvoie le vecteur $(0,\dots,0)$, ou n'importe quel vecteur $x=(x_1,\dots,x_n)$ tel que $x_1=0$, et à y non nul renvoie un vecteur x aléatoire a une probabilité de succès supérieure à $1/2$ avec une complexité constante.

Exercice 2, page 9. La borne du compromis temps-mémoire impose $2^m \geq \frac{t^2}{\varepsilon c^2} = \frac{2^{160}}{2^{-60}} = 2^{220}$. La sortie doit comporter plus de 220 symboles binaires.

Exercice 3, page 10. La simple application de la formule montre que le quotient $L(10^{200})/L(10^{300})$ vaut environ 16 000. Il faut donc 16 000 heures pour factoriser un entier de 300 chiffres, soit environ un an et dix mois.

Exercice 4, page 10. Non ! Considérer l'algorithme suivant :

Si n est pair, renvoyer le couple $(2, n/2)$, et dans le cas contraire, renvoyer un couple (p,q) aléatoire.

Dans trois cas sur quatre au cours du jeu FSU, au moins l'un des deux facteurs est pair et dans ce cas, cet algorithme renvoie un facteur.

Exercice 5, page 10. Soit e l'exposant public et d l'exposant privé. La quantité $ed-1$ multiple de $\mathrm{ppcm}(p-1,q-1)$ qui est pair. On peut donc écrire $ed-1=2^k u$ avec $k\geq 1$ et u impair. Soit a un élément quelconque de $\mathbb{Z}/n\mathbb{Z}$. Il vérifie $a^{ed-1}=(a^u)2^k=1$. Soit $b=a^u$. En élevant b au carré k fois, on obtient 1. L'avant dernière itération avant d'obtenir 1 fournit une racine carrée de 1. Si cette racine r carrée est différente de -1, alors $r^2-1=(r-1)(r+1)$ est multiple de n sans que, ni $r+1$, ni $r-1$ ne le soient. Le calcul de $\mathrm{pgcd}(n, r-1)$ fournit alors un facteur strict de n. Dans le cas contraire, recommencer avec une autre valeur de a.

Exercice 6, page 11. Si la factorisation de l'entier n est connue, soit $n=p\times q$. Rappelons que l'extraction des racines carrées modulo p a une solution efficace. Par exemple, si p est un entier premier qui est congru à 3 modulo 4, et si l'entier a est un un carré modulo p alors le calcul de $a^{(p+1)/4}$ fournit une racine carrée de a modulo p. Cette formule se généralise au cas où p n'est pas congru à 3 modulo 4 (par exemple l'algorithme de Shanks ou de Cipolla).

Pour extraire les racines carrées modulo n, il suffit d'extraire les racines modulo p et modulo q, et d'en déduire la racine carrée modulo n par application du théorème chinois des restes.

Réciproquement, supposons que l'on dispose d'un oracle qui calcule les racines carrées modulo n. L'algorithme suivant permet d'en déduire la factorisation de l'entier n :

Entrée : un entier n à factoriser.

1. Choisir un entier a au hasard dans $\mathbb{Z}/n\mathbb{Z}$.
2. Calculer $b=a^2$ et demander à l'oracle d'extraire la racine carrée de b. Soit r la valeur rendue par l'oracle.
3. Si $r=\pm a$, recommencer en 1.

4. $p = \text{pgcd}(n, a - r)$ fourni un facteur non trivial de n.

En effet, dans ce cas, $a^2 - b^2 = (a - r)(a + r)$ est multiple de n sans que $a - r$ ni $a + r$ ne le soient, les deux facteurs p et q de n se répartissent comme facteur premier de $a - r$ et $a + r$.

Exercice 7, page 15.

Exercice 8, page 16. Soit f une fonction à sens unique. Construisons à partir de f une fonction F à longueur régulière. Pour cela, on complète la valeur $f(x)$ par des zéros jusqu'à atteindre une longueur constante pour toutes les entrées de taille n. Comme f est à sens unique, elle est calculable efficacement et pour toute entrée de taille n, la taille de $f(x)$ est bornée par un polynôme en n. Soit $P(n)$ ce polynôme.

La fonction $F : x \longmapsto (f(x), 0^k)$, où $k = P(n) - |f(x)|$, est à longueur régulière par construction.

La fonction F est aussi à sens unique, car dans le cas contraire, il existe un algorithme polynomial qui l'inverse avec un succès non négligeable. Une modification mineure de cet algorithme permet d'inverser aussi f, ce qui contredit l'hypothèse selon laquelle f est à sens unique.

Exercice 9, page 16. Soit g une fonction à sens unique. La taille de $g(x)$ est bornée par un polynôme en la longueur de x, soit $|g(x)| \leq p(|x|)$, pour un polyôme p. On peut transformer la fonction g en une fonction à sens unique à longueur régulière en ajoutant des symboles aux valeurs et définir :

$$h(x) = g(x) \mid 10^{p(|x|)-|g(x)|-1}$$

Pour forcer la taille de $f(x)$ à être égale à celle x, considérer un paramètre de taille $p(|x|)$ et n'en prendre en compte que $|x|$ pour définir la valeur, en posant :

$$f(x \mid y) = h(x),$$

pour $|y| = p(|x|) - |x| + 1$. Il reste à montrer que la fonction ainsi construite est à sens unique. S'il existait un algorithme qui pour $z = f(y)$ produit un antécédent $x_1 \cdots x_n$, pour chaque entier $m \in \{1, \ldots, n\}$, on vérifie si $h(x_1 \cdots x_m) = y$.

Exercice 10, page 16.

1. Non ! Pas toujours ! Soit $E = F \times F$ et f_1 une fonction à sens unique $F \longrightarrow F$. On montre facilement que la fonction :

$$f : \begin{array}{ccc} E \times E & \longrightarrow & E \times E \\ (x_1, x_2) & \longmapsto & (f_1(x_2), 0) \end{array}$$

est à sens unique et vérifie $f \circ f : (x_1, x_2) \longmapsto (f_1(0), 0)$. Et une fonction constante n'est clairement pas à sens unique.

2. Oui. Montrons que si la fonction f est à sens unique, alors la fonction g l'est aussi. Pour cela, montrons la contraposée. Si la fonction g n'est pas à sens unique, alors la fonction f ne l'est pas non plus. On suppose qu'il existe un algorithme qui inverse g, et déduisons en un algorithme B qui inverse f.

   L'algorithme B prend en entrée un élément y de l'image de f.
   — Calculer $f(y)$. C'est possible car f est calculable dans le sens direct.
   — Soumettre le couple $(y, f(y))$ à l'algorithme A. L'algorithme A retourne un antécédent y par g qui est un antécédent de f.

   La probabilité de succès de l'algorithme B est la même que la probabilité de succès de l'algorithme A.

3. Oui. Comme à la question précédente, soit un algorithme A qui inverse $f \circ f$ et construisons un algorithme B qui inverse f.

   L'algorithme B prend en entrée un élément y de l'image de f. Il doit trouver l'unique antécédent de y par f.
   — Soumettre y à l'algorithme A. Soit x la valeur rendue par l'algorithme A qui est supposée être la valeur qui vérifie $f \circ f(x) = y$.
   — Rendre comme résultat $f(x)$ qui est par construction l'antécédent de y par f.

   L'algorithme B donne la bonne réponse avec une probabilité égale à celle de l'algorithme A.

   La différence avec la première question est que, comme f est une bijection, un élément aléatoire du domaine de f est aussi un élément aléatoire de l'image de f, ce qui est faux si f n'est pas bijective.

Exercice 11, page 17.

Exercice 12, page 18.

Exercice 13, page 18.

Exercice 15, page 18.

Exercice 16, page 19. Une fonction f constante n'est pas à sens unique, mais n'importe quel prédicat équilibré p est difficile. La valeur $f(x)$ n'apporte aucune information sur la valeur de x ni sur celle de de $p(x)$. Un adversaire n'a pas de meilleure stratégie que de tirer la valeur $b^\star$ au hasard.

Exercice 17, page 24. Soit $f : E \to F$ une fonction à sens unique injective et $p : E \to \{0,1\}$ un prédicat difficile de f. On peut par exemple prendre pour f la fonction RSA et pour p le chiffre binaire de parité.

Le joueur **A** choisit une valeur x aléatoire dans E. Sa valeur binaire est $\widehat{b} = p(x)$ et la valeur d'engagement est $c = f(x)$. Il peut aussi choisir la valeur binaire $\widehat{b}$ au hasard et répéter un choix au hasard de x jusqu'à ce que $p(x) = \widehat{b}$. La valeur d'ouverture est $d = x$.

La valeur d'ouverture est calculée à partir du couple (c, d) en vérifiant que $f(d) = c$, puis en révélant la valeur de $\widehat{b}$ par le calcul de $\widehat{b} = p(d)$.

Comme la fonction b est un prédicat difficile pour f, la valeur c ne révèle rien sur celle de $\widehat{b}$ et la propriété de dissimulation est satisfaite : il est pratiquement impossible de trouver b à partir de c.

Comme f est injective, il n'est pas possible de trouver deux valeurs distinctes d et d' qui ont la même valeur par f. La propriété d'enchérissement est donc satisfaite.

Exercice 1, page 28.

Exercice 2, page 31.

Exercice 3, page 31.

Exercice 5, page 32.

Exercice 6, page 32.

Exercice 7, page 32.

Exercice 4, page 31. Dans la formule 2.5 page 31, le second facteur est inféreur à 1 comme différence de deux probabilités. Ceci montre que l'avantage d'un adversaire D est inférieur à $1 - \Pr(y \in \mathrm{Im}(g))$, soit $1 - 1/2^k$ dans le cas présent.

Exercice 8, page 34.

Exercice 1, page 38.

Exercice 2, page 39.

Exercice 3, page 39. Non ! La famille RSA n'est pas une famille de fonctions pseudo-aléatoires, car les fonctions RSA sont multiplicatives. Si l'adversaire interroge l'oracle sur $f(x)$, $f(y)$ et $f(xy)$, et si les réponses vérifient $f(x)f(y) = f(xy)$, il est très probable que la fontion choisie par l'oracle soit une instance RSA. Une telle relation sur une fonction aléatoire est hautement improbable.

Exercice 4, page 40.

Exercice 5, page 40.

Exercice 6, page 43.

Exercice 7, page 43.

Exercice 1, page 47.

Exercice 2, page 47.

Exercice 3, page 48.

Exercice 4, page 48.

Exercice 5, page 51.

Exercice 6, page 51.

Exercice 7, page 51.

Exercice 1, page 55.

Bibliographie

[1] Katz (Jonathan) et Lindell (Yehuda) : *Introduction to Modern Cryptography.* Chapnan & Hall/CRC, 2014.

[2] Hellman (Martin) : A cryptanalytic time-memory trade-off. *IEEE Transaction on Information Theory*, 26(4):401–406, 1980.