Introduction à la Cryptographie

Notes de cours

André Schrottenloher

Inria
prenom.nom@inria.fr

Ces notes de cours suivent presque mot pour mot les notes d'Adeline Roux-Langlois et Léo Ackermann pour les précédentes itérations de ce cours, avec des erreurs en plus. Quelques parties ont été modifiées, d'autres le seront dans le futur.

Le matériel des TD a été volé à Adeline Roux-Langlois, Léo Ackermann, Nigel Smart (Cryptography, an introduction) et Boneh & Shoup (A graduate course in applied cryptography).

Introduction

L'objectif de ce cours est de donner une introduction très large aux concepts fondamentaux de la cryptographie moderne. Un accent particulier est mis sur les notions de sécurité calculatoire et parfaite (information-theoretic security). Nous introduirons les primitives fondamentales de la cryptographie asymétrique et symétrique, ainsi que des réductions de sécurité et des hypothèses de sécurité.

Ce cours se base en partie sur :

- Introduction to Modern Cryptography, Katz and Lindell
- Cryptography, an Introduction, Smart
- A graduate course in applied cryptography, Boneh and Shoup

Lorsque c'est nécessaire nous utilisons le **rouge** pour une information secrète, **bleu** pour une information qu'un attaquant pourrait écouter, et aucune couleur pour une information qui est publique.

Table des matières

1 Introduction à la Cryptographie

La **cryptographie** est une science qui s'intéresse à protéger l'**information** lorsqu'elle est transmise sur des canaux de transmission **non sécurisés**, en présence d'**adversaires** qui peuvent écouter ou modifier l'information transmise.

Elle emprunte des outils de différents domaines :

- La théorie de l'information ;
- La théorie de la complexité et les algorithmes ;
- Des probabilités, à la fois liées aux preuves de sécurité et aux algorithmes probabilistes ;
- La vérification, les systèmes de preuve formelle ;
- L'algèbre ;
- Et de nos jours, la théorie de l'information quantique.

1.1 Histoire

Si la cryptographie remonte bien plus loin (peut-être à l'invention de l'écriture), l'un des plus célèbres utilisateurs est certainement Jules César. L'historien romain Suetone (Suetonius) rapporte qu'il chiffrait des messages en déplaçant leurs lettres de trois positions dans l'alphabet Latin (le **chiffrement de César**). César lui-même rapporte dans son compte-rendu de la Guerre des Gaules (livre 5, chapitre 48) avoir utilisé un autre stratagème pour envoyer une lettre dans le camp encerclé de son général Cicéron (Cicero).

> *Tum cuidam ex equitibus Gallis magnis praemiis persuadet uti ad Ciceronem epistolam deferat. Hanc Graecis conscriptam litteris mittit, ne intercepta epistola nostra ab hostibus consilia cognoscantur. Si adire non possit, monet ut tragulam cum epistola ad amentum deligata intra munitionem castrorum abiciat.*

Ce qui se traduit :

> Then with great rewards he induces a certain man of the Gallic horse to convey a letter to Cicero. This he sends written in Greek characters, lest the letter being intercepted, our measures should be discovered by the enemy. He directs him, if he should be unable to enter, to throw his spear with the letter fastened to the thong, inside the fortifications of the camp.

En effet, la cryptographie historique se rapproche par bien des aspects de la linguistique : Alice et Bob n'essaient-ils pas, bon gré mal gré, de s'entendre sur un langage commun ne pouvant être compris que d'eux seuls ?

Cryptographie Moderne. C'est à la deuxième moitié du XXe siècle, avec l'avènement de l'informatique comme science, que la cryptographie bascule dans le giron de l'informatique fondamentale. Toutefois, certains principes fondateurs étaient déjà énoncés à la fin du XIXe siècle dans le travail d'Auguste Kerckhoffs [Ker83a ; Ker83b] :

1. *Le système doit être matériellement, sinon mathématiquement, indéchiffrable* ;
   La cryptographie moderne s'appuie plutôt sur le « matériellement » que « mathématiquement », comme nous le verrons avec la notion de sécurité calculatoire.

   Remarque 1. Diffie et Hellman [DH76] traduisirent ce principe en termes de classes de complexité : si le chiffrement et le déchiffrement sont des opérations en temps polynomial, le fait de trouver le secret (décrypter) est dans la classe NP. (En effet, la vérification consiste à re-chiffrer à l'aide du secret, et est donc en temps polynomial). Le problème doit être suffisamment difficile à l'intérieur de cette classe, donc idéalement NP-complet (en pratique, ce n'est pas le cas).

2. *Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi* ;
   En effet, la spécification d'un algorithme cryptographique doit être entièrement publique. Cela rejoint ce que nous expliquons au paragraphe « cryptanalyse » plus bas.
3. *La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants* ;
4. *Il faut qu'il soit applicable à la correspondance télégraphique* ;
5. *Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes* ;
6. *Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne de-mandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.*

Les quatre derniers principes concernent la **portabilité** et la **légèreté** du chiffrement. Kerckhoffs ne pouvait prévoir que ces tâches calculatoires, exécutées de son temps par des humains, puis des machines électromécaniques telles qu'Enigma, seraient aujourd'hui confiées à des processeurs. Toutefois, le problème des ressources reste fondamental. Pensons par exemple à :

- La **latence** : la moindre milliseconde supplémentaire de délai lors de la connexion à un site web représente une perte de parts de marché pour un navigateur web ou un fournisseur d'accès Internet ;
- **L'énergie** : de nombreux appareils fonctionnant sur batterie (Internet des objets) utilisent en permanence de la cryptographie ; chaque opération supplémentaire induit un coût en énergie et diminue leur durée de vie ;
- **La taille de circuit** : il n'y a pas beaucoup de place sur une carte à puce. Les algorithmes cryptographiques pouvant être utilisés (et les tailles de clés que l'on peut se permettre) y sont fortement contraints.

De manière plus générale, la cryptographie moderne recherche toujours une **minimisation des ressources calculatoires** (durée d'exécution, nombre d'opérations, taille des clés) pour un niveau de sécurité donné. Il n'est pas « très » difficile de faire de la cryptographie sûre, mais il est difficile de faire de la cryptographie sûre et efficace.

Cryptanalyse. Si la **cryptographie** vise à protéger l'information, la **cryptanalyse** vise à casser les constructions des cryptographes. Les deux se regroupent sous le terme de **cryptologie** (mais on parle souvent de **cryptographie** en ce sens, donc ne soyez pas surpris).

L'ouvrage le plus ancien de cryptanalyse est celui d'Al-Kindi, un scientifique et philosophe arabe né en 801. Dans « Un manuscrit sur le décryptage de messages cryptographiques », il invente la **cryptanalyse statistique** des chiffrements par substitution tels que celui de César, basée sur l'analyse fréquentielle des lettres du texte chiffré, ainsi que les attaques à mot probable [Sin99 ; Al-92].

Remarque 2. Notez que pour attaquer le chiffrement de César, il n'est pas nécessaire de procéder à l'analyse fréquentielle : il n'y a que 26 clés possibles, ce qui peut être testé à la main. Si la force brute suffit, on l'utilise. En revanche le nombre de clés possible est beaucoup plus élevé pour une permutation quelconque de l'alphabet, d'où l'analyse fréquentielle.

La conception et la cryptanalyse sont deux facettes indissociables de la cryptographie moderne. Toutes deux sont menées en parallèle, et les allers-retours sont nombreux avant de parvenir à concevoir des standards de chiffrement pouvant être considérés comme sûrs.

1.2 Exemples

La cryptographie à usage commercial se trouve partout dans notre monde numérique. Pensez par exemple au protocoles de communications sur les réseaux (HTTPS, SSL, TLS, PGP, réseaux wifi, téléphones mobiles, ...), aux implémentations matérielles (cartes de crédit, DVD, Blu-ray, Passe Navigo), aux logiciels tels que les DRM. Historiquement, on pense principalement au **chiffrement**, mais la cryptographie offre de très nombreuses autres fonctionnalités : signature numérique, calcul multipartite sécurisé, preuves de connaissance, etc.

Exemple : Protocole TLS. Soit un client **Alice** (mon ordinateur) et un serveur **Bob** (le serveur de ma banque). Le protocole TLS permet à Alice et Bob de mettre en place une connexion sécurisée. Il y a globalement deux étapes :

1. Alice et Bob s'authentifient l'un à l'autre et s'accordent sur une clé de chiffrement partagée. Cette étape requiert des outils de **cryptographie asymétrique**.
2. Alice et Bob communiquent en chiffrant leurs messages à l'aide de la clé partagée. Cette étape requiert des outils de **cryptographie symétrique**.

Les informations échangées étant extrêmement sensibles, le protocole se doit d'offrir des **garanties de sécurité** :

- **Authenticité** : Alice doit être sûre qu'elle parle à Bob
- **Confidentialité** : les données échangées ne doivent pas pouvoir être lues par une entité qui observerait tous nos échanges
- **Intégrité** : les données ne doivent pas pouvoir être modifiées par cette entité

Plus exactement, on ne peut pas **empêcher** l'adversaire **Eve** de modifier les paquets échangés, mais ce doit être alors détecté (et le protocole doit prévoir un arrêt dans ce cas).

C'est un cas typique, mais il y a beaucoup d'autres situations qui nécessitent d'autres garanties. Par exemple, pour le vote électronique, nous avons besoin de compter les bulletins tout en préservant le secret et l'anonymat d'un bulletin quelconque donné. C'est possible, à l'aide d'un protocole adapté.

1.3 Conception de la Cryptographie

La cryptographie est un jeu de construction où nous partons d'objets simples pour arriver à des objets compliqués.

- Les **primitives** sont des briques de base offrant une fonctionnalité de bas niveau (ex. : chiffrement à bloc, chiffrement à clé publique, KEM, ...). Pensons par exemple au chiffrement symétrique : il ne permet que de chiffrer et ne garantit que la confidentialité.
- Les **modes d'opération** (spécifiques à la cryptographie symétrique) sont des manières génériques de fabriquer une primitive à partir d'une autre primitive encore plus simple, par exemple un chiffrement à flot à partir d'un chiffrement à bloc.
- Les **protocoles** font un usage en « boîte noire » d'un certain nombre de primitives. Ils spécifient une série de communications permettant d'arriver à une certaine fonctionnalité (« communication sûre ») et des propriétés de sécurité avancées. L'usage des primitives en « boîte noire » signifie qu'elles peuvent être instanciées par n'importe quelle primitive satisfaisant les notions de sécurité exigées.

Exemple 1. Dans TLS, vous avez besoin d'une fonction de hachage. Vous pouvez utiliser n'importe quelle fonction de hachage **cryptographique**. Mais pas n'importe quelle fonction de hachage. En effet, nous avons besoin de certaines propriétés de sécurité.

Sécurité d'un Protocole. La conception d'un protocole demande de définir précisément à quoi il sert et quelles notions de sécurité il doit satisfaire. C'est souvent une étape délicate. En effet, par essence, cette définition de sécurité est une **modélisation** de la vie réelle : l'attaquant réel est modélisé par un adversaire mathématique (une machine de Turing) capable d'accéder à diverses informations (par exemple les messages échangés) et essayant de résoudre un certain problème (par exemple trouver la clé secrète).

Il est tout à fait possible que le protocole soit sûr pour la définition que nous avons choisie, mais que cette définition ne soit pas suffisante pour éviter certaines attaques auxquelles nous n'aurions pas pensé, et qui seraient dévastatrices en pratique (nous aurions pu modéliser un adversaire qui écoute seulement, sans penser qu'il peut aussi modifier les messages échangés et les remplacer par les siens).

Nous écrivons ensuite une **preuve** qui **réduit** la sécurité du protocole à celle des primitives. Cela signifie que s'il existe une attaque qui casse la propriété de sécurité, alors il existe une attaque qui casse l'une des propriétés de sécurité des primitives.

Sécurité des Primitives. La sécurité des primitives repose (le plus souvent) sur des hypothèses calculatoires. Ces hypothèses diffèrent beaucoup entre la cryptographie symétrique et la cryptographie asymétrique.

En cryptographie asymétrique, on arrive souvent à réduire la sécurité à un problème mathématique bien défini (problème RSA, problème DDH...). Cela signifie qu'on a une nouvelle réduction : s'il existe un adversaire capable de casser la primitive, alors il existe un algorithme efficace pour ce problème mathématique.

En cryptographie symétrique, on se retrouve la plupart du temps avec des hypothèses de sécurité *ad hoc*, du type : « voici mon schéma de chiffrement, on n'a pas trouvé d'attaque dessus ».

Remarque 3. Il n'est en réalité pas impossible de baser la cryptographie symétrique sur des problèmes mathématiques tels que la factorisation. Mais ce serait surtout inutilement dispendieux (et on veut être efficace !).

Il n'y a pas de hiérarchie des hypothèses de sécurité. Dans les deux cas, puisque nous ne sommes pas capables de **prouver** mathématiquement la sécurité, elle est prouvée **empiriquement** en cherchant des attaques. Dans les deux cas, ces attaques sont des algorithmes visant à résoudre des problèmes précis.

De la conception aux standards, il y a plusieurs étapes :

1. Des personnes conçoivent une primitive / un protocole ;
2. Iels font une analyse de sécurité ;
3. Iels le publient en faisant des affirmations de sécurité (*security claim*) précises : pas seulement « nous pensons que c'est sûr », mais « nous pensons que la sécurité est de 128 bits pour tout adversaire limité à 2^{64} bits de mémoire dans le modèle IND-CCA2 ... »
4. Le reste du monde essaie de contredire ces affirmations : « nous avons trouvé une attaque en temps 2^{127} avec une mémoire limitée à 2^{63} bits ». Ou au contraire : « nous avons trouvé une attaque, mais uniquement dans un modèle affaibli ; il semble que la primitive complète soit sûre comme prédit par les auteur·ices. »
5. Au bout d'un moment, le nouvel algorithme pourra être adopté dans le portfolio de standards d'une institution comme l'ISO ou l'IETF

Toutes ces étapes sont importantes. Nous avons besoin :

- D'une explication claire des choix de conception, afin de s'assurer qu'il ne contiennent pas de porte dérobée ;
- D'une analyse de sécurité étendue et à jour avec l'état de l'art de la cryptanalyse, **et qui se poursuit dans le temps** selon le principe suivant :

$$\text{Confiance } = \int_{t=0}^{+\infty} \text{Effort de cryptanalyse } \; dt \tag{1}$$

Bien sûr, tout ne se déroule pas toujours de cette manière. Parfois, en contradiction totale avec les principes de Kerckhoffs, des consortiums ou industriels utilisent leurs propres standards secrets, comme la suite de chiffrement du GSM[Bei+21] ; ces standards sont utilisés des décennies avant que leur spécification ne soit divulguée et qu'on se rende compte qu'ils n'étaient pas sûrs.

1.4 Chiffrement Symétrique

Définition 1. Un schéma de chiffrement symétrique est un triplet d'algorithmes KeyGen, Enc, Dec opérant sur trois ensembles finis : l'espace des clés $\mathcal{K}$, l'espace des messages (textes clairs) $\mathcal{M}$, l'espace des chiffrés $\mathcal{C}$, avec la signature suivante :

$$\begin{cases} \mathsf{KeyGen} \; : \emptyset \to \mathcal{K} \\ \mathsf{Enc} \; : \mathcal{K} \times \mathcal{P} \to \mathcal{C} \\ \mathsf{Dec} \; : \mathcal{K} \times \mathcal{C} \to \mathcal{M} \end{cases} \tag{2}$$

et la propriété (dite de **correction**) :

$$\forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m \; . \tag{3}$$

Dans la suite nous supposons que tous les chiffrés de $\mathcal{C}$ sont **accessibles**, c'est-à-dire que pour tout chiffré c dans $\mathcal{C}$, il existe un message m dans $\mathcal{M}$ et une clé k dans $\mathcal{K}$ tels que $\mathsf{Enc}(k, m) = c$.

Exemple 2. L'identité : $\mathsf{Enc}(k, m) = m$ and $\mathsf{Dec}(k, m) = m$ est un chiffrement valide selon la définition. Mais naturellement, il n'est pas sûr.

En cryptographie symétrique la fonction KeyGen, qui ne prend aucun argument et renvoie une clé $k \in \mathcal{K}$, est souvent triviale : on se contente de choisir une clé uniformément au hasard dans $\mathcal{K}$: $k \hookleftarrow U(\mathcal{K})$.

Dans tous les cas, d'après les principes de Kerckhoffs les algorithmes KeyGen, Enc et Dec eux-mêmes sont publiés. Le secret n'est localisé que dans la clé k.

Définition 2 (Sécurité parfaite (Shannon 49))**.** Un chiffrement symétrique sur l'espace des messages $\mathcal{M}$ est parfaitement sûr si, pour toute variable aléatoire M à valeurs dans $\mathcal{M}$, tout message $m \in \mathcal{M}$ et chiffré $c \in \mathcal{C}$:

$$\Pr\left[M = m | \mathsf{Enc}(\mathsf{KeyGen}, M) = c\right] = \Pr\left[M = m\right] \; . \tag{4}$$

C'est la notion fondamentale de la **sécurité parfaite**, parfois appelée *information-theoretic security.*

- L'adversaire connaît la distribution du message M
- L'adversaire observe le chiffré $\mathsf{Enc}(\mathsf{KeyGen}, M)$ qui a été construit en choisissant une clé selon la spécification de KeyGen (par exemple $k \hookleftarrow U(\mathcal{K})$)
- Iel est incapable de déduire la moindre information supplémentaire sur le message. En d'autres termes, les variables aléatoires M et $\mathsf{Enc}(\mathsf{KeyGen}, M)$ sont indépendantes.

Cette définition est équivalente à la suivante.

Définition 3. Un chiffrement symétrique est parfaitement sûr si, pour tous $m_1, m_2, c \in \mathcal{M} \times \mathcal{M} \times \mathcal{C}$:

$$\Pr_{k \leftarrow \mathsf{KeyGen}}\left[\mathsf{Enc}(k, m_1) = c\right] = \Pr_{k \leftarrow \mathsf{KeyGen}}\left[\mathsf{Enc}(k, m_2) = c\right] \; . \tag{5}$$

Lemme 1. *La Définition 2 et la Définition 3 sont équivalentes.*

▷ **Démonstration en TD.**

Démonstration. **Première implication.** Supposons que pour tous m_1, m_2, c :

$$\Pr_{k\leftarrow \mathsf{KeyGen}}[\mathsf{Enc}(k, m_1) = c] = \Pr_{k\leftarrow \mathsf{KeyGen}}[\mathsf{Enc}(k, m_2) = c]$$

On a, pour toute variable aléatoire M, m et c :

$$\begin{aligned}\Pr[M = m|\mathsf{Enc}(\mathsf{KeyGen}, M) = c] &= \frac{\Pr[M = m \wedge \mathsf{Enc}(\mathsf{KeyGen}, M) = c]}{\Pr[\mathsf{Enc}(\mathsf{KeyGen}, M) = c]} \\ &= \frac{\Pr[M = m \wedge \mathsf{Enc}(\mathsf{KeyGen}, m) = c]}{\Pr[\mathsf{Enc}(\mathsf{KeyGen}, M) = c]} \\ &= \Pr[M = m]\,\frac{\Pr[\mathsf{Enc}(\mathsf{KeyGen}, m) = c]}{\Pr[\mathsf{Enc}(\mathsf{KeyGen}, M) = c]}\end{aligned}$$

On montre que :

$$\begin{aligned}\Pr[\mathsf{Enc}(\mathsf{KeyGen}, M) = c] &= \sum_{m'\in\mathcal{M}} \Pr[M = m']\Pr[\mathsf{Enc}(\mathsf{KeyGen}, m) = c] \\ &= \sum_{m'\in\mathcal{M}} \Pr[M = m']\left(\Pr[\mathsf{Enc}(\mathsf{KeyGen}, m) = c]\right) \text{ par hypothèse} \\ &= \Pr[\mathsf{Enc}(\mathsf{KeyGen}, m) = c] \underbrace{\sum_{m'\in\mathcal{M}} \Pr[M = m']}_{=1}\end{aligned}$$

Donc :

$$\Pr[M = m|\mathsf{Enc}(\mathsf{KeyGen}, M) = c] = \Pr[M = m] \quad .$$

Deuxième implication. Supposons que pour toute variable aléatoire M, m et c :

$$\Pr[M = m|\mathsf{Enc}(\mathsf{KeyGen}, M) = c] = \Pr[M = m] \quad .$$

Soit m_1, m_2, c et M la distribution uniforme sur $\{m_1, m_2\}$. On a :

$$\begin{cases}\Pr[M = m_1|\mathsf{Enc}(\mathsf{KeyGen}, M) = c] = \Pr[M = m_1] = \frac{1}{2} \\ \Pr[M = m_2|\mathsf{Enc}(\mathsf{KeyGen}, M) = c] = \Pr[M = m_2] = \frac{1}{2}\end{cases}$$

Donc :

$$\begin{gathered}\Pr[M = m_1|\mathsf{Enc}(\mathsf{KeyGen}, M) = c] = \Pr[M = m_2|\mathsf{Enc}(\mathsf{KeyGen}, M) = c] \\ \frac{\Pr[M = m_1]\Pr[\mathsf{Enc}(\mathsf{KeyGen}, m_1) = c]}{\Pr[\mathsf{Enc}(\mathsf{KeyGen}, M) = c]} = \frac{\Pr[M = m_2]\Pr[\mathsf{Enc}(\mathsf{KeyGen}, m_2) = c]}{\Pr[\mathsf{Enc}(\mathsf{KeyGen}, M) = c]} \\ \Pr[\mathsf{Enc}(\mathsf{KeyGen}, m_1) = c] = \Pr[\mathsf{Enc}(\mathsf{KeyGen}, m_2) = c] \quad .\end{gathered}$$

□

1.5 One-Time Pad (Chiffrement de Vernam)

Définition 4 (One-time Pad)**.** Soit $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^n$ pour $n > 0$. Le chiffrement *One-time Pad* est défini ainsi :

$$\begin{cases}\mathsf{KeyGen} & : k \xleftarrow{\$} \mathcal{K} \\ \mathsf{Enc} & : m \mapsto m \oplus k \\ \mathsf{Dec} & : c \mapsto c \oplus k\end{cases} \tag{6}$$

Il est correct :

$$\forall k, m, \mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m \oplus k \oplus k = m \ . \tag{7}$$

Lemme 2. *Le One-Time Pad est parfaitement sûr.*

Démonstration. Cela vient de la propriété du XOR. Si l'on XOR une variable aléatoire uniforme dans $\{0,1\}$ (bit à bit) à une valeur quelconque, on obtient de nouveau une variable aléatoire uniforme.

$$\begin{aligned}\Pr\left[\mathsf{Enc}(K, M) = c | M = m\right] &= \Pr\left[M \oplus K = c | M = m\right] \\ &= \Pr\left[m \oplus K = c\right] \\ &= \Pr\left[K = m \oplus c\right] = 2^{-n}\end{aligned}$$

□

Le One-time Pad a eu son heure de gloire dans l'histoire de la crypto, mais son principal désavantage se trouve dans son titre : la clé ne doit être utilisée qu'une seule fois. En effet, à partir d'une paire clair / chiffré on peut retrouver la clé.

De plus, le fait d'avoir une clé aussi longue que le message n'est pas envisageable en pratique. Comment transmettre une clé aussi longue ? Il vaudrait mieux pouvoir partir d'une petite clé, que l'on pourrait « étendre » en une séquence pseudo-aléatoire de grande taille : c'est précisément ce que permet un chiffrement à flot.

1.6 Théorème de Shannon

Lemme 3. *La sécurité parfaite implique* $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{M}|$.

▷ **Démonstration en TD.**

Démonstration. La fonction $\mathsf{Enc}(k, \cdot)$ se doit être injective pour tout $k \in \mathcal{K}$, donc $|\mathcal{C}| \geq |\mathcal{M}|$.

Si nous fixons $m \in \mathcal{M}$, alors par la propriété de sécurité parfaite $\Pr[C = c | M = m] = \Pr[C = c] > 0$ pour tout c. Donc pour tout m il doit exister une clé qui envoie m sur c, i.e., $|\mathcal{K}| \geq |\mathcal{C}|$. □

Cela implique que le One-time Pad est une solution optimale, mais aussi, qu'aucun chiffrement avec des clés plus petites ne peut offrir de sécurité parfaite. C'est pourquoi on se rabat sur la notion de **sécurité calculatoire** (computational secrecy).

Théorème 1 (Théorème de Shannon)**.** *Soit* $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ *un schéma de chiffrement symétrique sur* $\mathcal{K}, \mathcal{M}, \mathcal{C}$ *tel que* $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$*. Il offre la sécurité parfaite si et seulement si :*

1. *Chaque clé est choisie avec probabilité* $1/|\mathcal{K}|$ *;*
2. *Pour tout* $m \in \mathcal{M}, c \in C$*, il existe une unique* k *telle que* $c = \mathsf{Enc}(m, k)$*.*

▷ **Démonstration en TD.**

Résumé

- On dit **chiffrer**, **déchiffrer**. **Décrypter** veut dire casser le schéma. Tout autre terme (crypter, encrypter…) est à proscrire.
- La cryptographie vise à assurer la sécurité de l'information en présence d'un **adversaire actif**, notamment sa confidentialité, intégrité, et authenticité
- Pour un chiffrement, la sécurité parfaite est assurée par le One-time Pad, mais inenvisageable en pratique.
- Pour assurer sa sécurité, quel qu'il soit, un schéma cryptographique doit être un **algorithme public** et faire l'objet d'une analyse de sécurité externe dans la durée.

Références

[Al-92] Ibrahim A. Al-Kadit. “Origins of cryptology : The Arab contributions”. In : *Cryptologia* 16.2 (1992), p. 97-126.

[Bei+21] Christof Beierle, Patrick Derbez, Gregor Leander, Gaëtan Leurent, Håvard Raddum, Yann Rotella, David Rupprecht et Lukas Stennes. “Cryptanalysis of the GPRS Encryption Algorithms GEA-1 and GEA-2”. In : *EUROCRYPT (2)*. T. 12697. Lecture Notes in Computer Science. Springer, 2021, p. 155-183.

[DH76] Whitfield Diffie et Martin E. Hellman. “New directions in cryptography”. In : *IEEE Trans. Inf. Theory* 22.6 (1976), p. 644-654. DOI : 10.1109/TIT.1976.1055638. URL : https://doi.org/10.1109/TIT.1976.1055638.

[Ker83a] Auguste Kerckhoffs. “La cryptographie militaire [Military cryptography, part 1]”. In : *Journal des sciences militaires [Military Science Journal]* (January 1883). URL : https://www.petitcolas.net/kerckhoffs/crypto_militaire_1_b.pdf.

[Ker83b] Auguste Kerckhoffs. “La cryptographie militaire [Military cryptography, part 2]”. In : *Journal des sciences militaires [Military Science Journal]* (February 1883). URL : https://www.petitcolas.net/kerckhoffs/crypto_militaire_2.pdf.

[Sin99] Simon Singh. *The code book : the secret history of codes and codebreaking*. Fourth Estate London, 1999.

2 Définir la Sécurité

La sécurité du One-Time Pad (vu au cours précédent) est dite **parfaite** (ou *information-theoretic*). Même si l'adversaire dispose d'une puissance de calcul illimitée, iel ne dispose tout simplement de pas assez d'information pour retrouver le clair et / ou la clé secrète.

Malheureusement, dans le contexte du chiffrement symétrique, on ne peut pas utiliser un tel schéma en pratique. (Dans d'autres situations, la sécurité parfaite revient sur scène, par exemple le protocole de partage de secrets de Shamir, mais ce ne sera pas étudié dans ce cours). C'est pourquoi la cryptographie moderne repose presque exclusivement sur la notion, plus faible, de **sécurité calculatoire**.

2.1 Notion d'Adversaire et Temps de Calcul

Souvenons-nous qu'un schéma cryptographique fait intervenir des **participants** légitimes (Alice, Bob) et un **adversaire** (Eve). Tous sont des algorithmes (i.e., machines de Turing). L'adversaire dispose d'un accès à certaines facettes du schéma, par exemple les messages échangés par Alice et Bob.

Remarquons pour commencer que :

- Un adversaire qui réussit doit être **efficace** en temps. Typiquement, un algorithme **efficace** termine en temps polynomial (ce doit être aussi le cas des parties légitimes dans un protocole cryptographique !). De bonnes preuves de sécurité sont capables de donner des bornes assez fines sur le temps de calcul d'un adversaire.
- Un adversaire est un algorithme **probabiliste**. Par exemple, il peut réussir en devinant la clé au hasard. Nous devons accepter ce fait : il aura toujours une certaine probabilité de succès ; l'objectif est que cette probabilité soit **négligeable** lorsque l'adversaire est en temps polynomial.

Exemple 3. Considérons une clé secrète de n bits (soit 2^n possibilités) : un adversaire pourrait deviner la clé en tirant au hasard et réussirait alors avec probabilité 2^{-n}. Ou bien, il pourrait essayer toutes les clés et réussirait alors avec probabilité 1, mais en temps 2^n (qui est exponentiel).

Définition 5. Un schéma cryptographique est (t, ε)-sûr si tout adversaire exécutant en temps t réussit à attaquer le schéma avec probabilité au plus ε.

Notons qu'à ce stade nous n'avons pas encore défini ce que signifie « attaquer » le schéma. Nous parlerons aussi de **casser** une propriété de sécurité.

Définition Asymptotique. Dans le cadre de ce cours, nous nous plaçons dans un contexte asymptotique (plus proche en esprit du cadre de la cryptographie à clé publique). Dans ce contexte le temps de calcul de l'adversaire et sa probabilité de succès sont des fonctions d'un paramètre de sécurité noté λ ou n (pour nous, ce sera n).

- Un algorithme est **polynomial** en n s'il existe une constante c telle que son temps de calcul est $\mathcal{O}(n^c)$. Notez que comme nous considérons a priori des algorithmes probabilistes, nous avons une certaine souplesse dans cette définition : nous autorisons un algorithme de type « Las Vegas » qui réussit toujours mais termine en temps $\mathcal{O}(n^c)$ **avec probabilité constante**.
- Une probabilité de succès est **négligeable** si pour toute constante c, elle est en $o(n^{-c})$ (c'est-à-dire plus petite que l'inverse de tout polynôme).

Un algorithme efficace sera aussi qualifié de **PPT** (*probabilistic polynomial-time*).

Notez quelques propriétés :

$$\begin{cases} \text{negl} + \text{negl} = \text{negl} \\ \text{negl} \cdot \text{poly} = \text{negl} \\ \text{poly} \cdot \text{poly} = \text{poly} \end{cases} \tag{8}$$

Le cas typique ce que qui n'est **pas** polynomial est une complexité exponentielle en 2^n ; mais une complexité en $2^{\mathcal{O}(\sqrt{n})}$ (dite **sous-exponentielle**) est aussi hors du domaine de l'« efficace » . Du côté des fonctions négligeables, une exponentielle 2^{-n} est typique, mais une fonction qui décroît moins vite comme $1/n^{\log n}$ est aussi considérée comme négligeable.

Remarque 4. Si l'adversaire peut attaquer le schéma en temps polynomial, cela signifie que l'usage légitime (qui est $\text{poly}(n)$) et le décryptage (qui serait donc $\text{poly}(n)$) sont tous deux dans la même classe de complexité. En d'autres termes il n'y a pas de véritable différence entre leur difficulté, et le schéma ne sert à rien.

Quelques Nombres. Comme nous parlons de sécurité calculatoire, nous avons besoin de mettre quelques chiffres sur les nombres d'opérations considérés comme élevés ou infaisables. Par opération, pensez à un cycle CPU. (Ce n'est pas une opération sur un bit : une addition d'un entier de 32 bits ne demande par exemple qu'un seul cycle).

Nous avons $1000 \simeq 2^{10}$. Un CPU à 4GHz exécute $4 \times 10^9 \simeq 2^{32}$ opérations par seconde. Mon ordinateur portable a $8 = 2^3$ cœurs, et le calcul peut durer des heures (une heure $\simeq 2^{12}$ secondes). Donc un nombre d'opérations de l'ordre de 2^{40} est faisable pour un seul ordinateur.

Songez que les GPUs permettent une parallélisation massive sur des milliers de cœurs, donc ajoutons un ou deux facteurs 2^{10} par sécurité : un nombre d'opérations de l'ordre de 2^{60} est faisable dans l'année pour les chercheurs académiques (avec un peu de financement).

Le réseau bitcoin calcule de l'ordre de 2^{90} fonctions de hachage par an. Ces calculs sont massivement distribués, et exécutés à l'aide de matériel spécialisé (des ASICs, *Application-Specific Integrated Circuit*, des circuits dont le seul objectif est de calculer un algorithme précis). C'est aussi le niveau de puissance calculatoire d'un adversaire *state-level* motivé.

Toutefois, pour cette puissance de calcul le réseau Bitcoin utilisait également récemment 130 TWh d'énergie par an, ce qui est assez pour vaporiser 2.07×10^{11} litres d'eau. En extrapolant pour calculer 2^{128} hachés, on aurait besoin de $2^{38} \simeq 2.75 \times 10^{11}$ fois plus d'énergie, soit l'équivalent de vaporiser 5.7×10^{22} litres d'eau. C'est plus que toute l'eau des océans de la Terre (environ 1.332×10^{21} litres). En d'autres termes, bien que 2^{90} opérations soit de l'ordre du faisable, 2^{128} est physiquement infaisable en l'état actuel de notre matériel informatique.

Quant aux probabilités de succès, une probabilité de l'ordre de 2^{-32} correspond à une chance sur quatre milliards. Est-ce rare ? Une personne ne se connecte pas un milliard de fois à un site web, mais il y a tous les jours des milliards de connections TLS. Une probabilité 2^{-64} devient déjà plus improbable. Si un événement arrive avec probabilité 2^{-64} chaque seconde, c'est une fois tous les 400 milliards d'années.

2.2 Chiffrement à Clé Publique

La notion de chiffrement à clé publique a été introduite par Diffie et Hellman dans un article de 1976 “New directions in Cryptography” [DH76]. Ses premières réalisations ont vu le jour peu après avec les cryptosystèmes de McEliece, ElGamal et le RSA (Rivest, Shamir et Adelman) qui est devenu l'algorithme le plus utilisé.

L'idée est de créer une asymétrie entre l'information nécessaire pour chiffrer (clé publique) et pour déchiffrer (clé privée ou clé secrète). C'est l'équivalent d'un cadenas que tout le monde peut fermer mais qui nécessite une clé pour être ouvert.

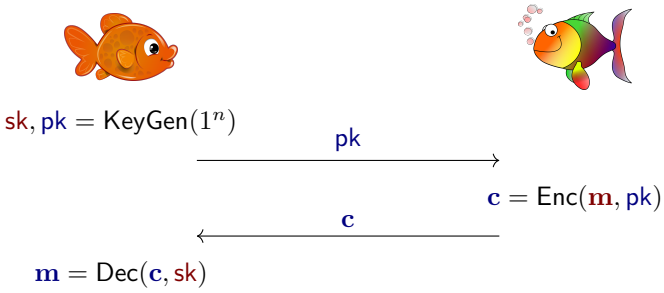
Définition 6 (PKE)**.** Un schéma de chiffrement à clé publique (*PKE = Public-Key Encryption Scheme*) est un triplet d'algorithmes polynomiaux :

$$\begin{cases} \mathsf{KeyGen} : & 1^n & \mapsto & \mathsf{sk}, \mathsf{pk} \\ \mathsf{Enc} : & \mathbf{m}, \mathsf{pk} & \mapsto & \mathbf{c} \\ \mathsf{Dec} : & \mathbf{c}, \mathsf{sk} & \mapsto & \mathbf{m} \end{cases} \tag{9}$$

tels que pour tout m, $\mathsf{Dec}(\mathsf{sk}, (\mathsf{Enc}(\mathsf{pk}, m), m)) = m$.

Remarque 5. Notez la pédanterie consistant à passer le paramètre de sécurité n sous la forme d'une chaîne de caractères unaire $1^n = 1 \cdots 1$, afin de respecter la règle d'un algorithme « polynomial ». Si n était passé sous forme de nombre le KeyGen ne serait plus forcément un algorithme polynomial en la taille de son entrée.

Alice utilise le KeyGen pour générer une paire de clés, et distribue la clé publique. Bob utilise la clé publique pour chiffrer le message $\mathbf{m}$. Alice utilise la clé secrète pour déchiffrer le message $\mathbf{c}$.



2.3 Indistinguabilité

Indistinguabilité Calculatoire. Deux distributions de probabilité sont **indistinguables calculatoirement** (*computationally indistinguishable*) si aucun algorithme efficace ne peut les distinguer ; i.e., ayant accès à des échantillons (*samples*) d'une distribution, décider de laquelle il s'agit.

Nous formalisons cette notion en considérant un **distingueur**, qui est simplement un algorithme probabiliste placé dans le contexte de deux **jeux**. Ces jeux se déroulent comme une mauvaise blague :

> Ce sont deux algorithmes qui discutent, l'un renvoie 0 ou 1, fin.

Définition 7 (Distingueur)**.** Soit D_0, D_1 deux distributions sur $\{0,1\}^n$. Les jeux G_0 et G_1 sont définis ainsi : l'adversaire $\mathcal{D}$ échange avec un *challenger* $\mathcal{C}$.

Au cours du jeu, l'adversaire peut faire une requête : le challenger renvoie alors un élément tiré selon la distribution D_b : $x \hookleftarrow D_b$, où $D_b = D_0$ dans le jeu G_0, et $D_b = D_1$ dans le jeu G_1. À la fin, $\mathcal{D}$ renvoie un bit b'.

L'**avantage** du distingueur $\mathcal{D}$ est défini par la différence entre les probabilités de renvoyer 1 dans les deux jeux :

$$\mathrm{Adv}(\mathcal{D}) = |\Pr\left[\mathcal{D} \xrightarrow{G_0} 1\right] - \Pr\left[\mathcal{D} \xrightarrow{G_1} 1\right]| \ . \tag{10}$$

$\mathcal{D}$ est un **distingueur** si $\mathrm{Adv}(\mathcal{D}) \geq \varepsilon$ pour un ε non-négligeable en n.

Notez bien que l'avantage est une différence de probabilité dans les deux jeux (deux « expériences »). Cela indique la différence de comportement de l'adversaire dans les deux cas. Si l'adversaire a le même comportement, alors ce n'est pas un distingueur.

Il existe une autre définition faisant intervenir un seul jeu.

Définition 8 (Distingueur)**.** Soit D_0, D_1 deux distributions sur $\{0,1\}^n$.

On définit un jeu G entre l'adversaire $\mathcal{D}$ et le challenger $\mathcal{C}$ qui se déroule en trois phases.

1. **Initialisation** : le challenger choisit un bit $b \in \{0,1\}$ uniformément au hasard : $b \hookleftarrow U(0,1)$.
2. **Requêtes** : sur requête de l'adversaire, le challenger renvoie un élément tiré selon la distribution D_b : $x \hookleftarrow D_b$ (cela peut arriver un nombre quelconque de fois)
3. **Finalisation** : $\mathcal{D}$ renvoie un bit b' à $\mathcal{C}$. Si $b = b'$ on renvoie "Win", sinon "Lose".

$\mathcal{D}$ est un **distingueur** s'il existe une fonction non-négligeable ε telle que $\Pr[Win] \geq 1/2 + \varepsilon$ dans le jeu G.

Remarque 6. L'adversaire a droit à autant de requêtes qu'il le souhaite, mais bien sûr il y a une limite à cela : il doit s'agir d'un algorithme polynomial.

Définition 9 (Indistinguabilité)**.** Deux distributions D_0 et D_1 sont calculatoirement indistinguables si pour tout distingueur PPT $\mathcal{D}$ il existe une fonction négligeable negl telle que :

$$|\Pr\left[\mathcal{D} \xrightarrow{G_0} 1\right] - \Pr\left[\mathcal{D} \xrightarrow{G_1} 1\right]| \leq \text{negl}(n) \tag{11}$$

De manière équivalente, deux distributions sont indistinguables si pour tout distingueur PPT $\mathcal{D}$, il existe une fonction négligeable negl telle que :

$$|\Pr[Win] - 1/2| \leq \text{negl}(n) \tag{12}$$

Démonstration. Nous prouvons l'équivalence de ces deux définitions à l'aide d'une double réduction. Dans les deux cas, on suppose l'existence d'un distingueur (donc un algorithme) pour une définition et on s'en sert pour construire un distingueur (donc un autre algorithme) pour l'autre définition.

Soit $\mathcal{A}$ un distingueur selon la Définition 8. Soit $\mathcal{A}'$ qui agit exactement comme $\mathcal{A}$ mais renvoie b' et termine au lieu d'envoyer b' à $\mathcal{C}$. Alors :

$$\begin{aligned}
\text{Adv}(\mathcal{A}') &= \Pr[b' = 1 | b = 0] - \Pr[b' = 1 | b = 1] \\
&= |1 - \Pr[b' = 0 | b = 0] - \Pr[b' = 1 | b = 1]| \\
&= |1 - 2\Pr[b' = 0 \wedge b = 0] - 2\Pr[b' = 1 \wedge b = 1]| \\
&= |1 - 2\Pr[Win]| \geq 2\varepsilon \ .
\end{aligned}$$

Inversement, considérons un distingueur $\mathcal{A}'$ de la Définition 7. Soit $\mathcal{A}$ tel qu'il agit comme $\mathcal{A}'$, mais envoie b' à $\mathcal{C}$. Alors :

$$\begin{aligned}
\Pr[Win] &= \Pr[b' = 0 \wedge b = 0] + \Pr[b' = 1 \wedge b = 1] \\
&= \frac{1}{2}(\Pr[b' = 0 | b = 0] + \Pr[b' = 1 | b = 1]) \\
&= \frac{1}{2}(1 + \text{Adv}(\mathcal{A}'))
\end{aligned}$$

En supposant que $\Pr[b' = 1 | b = 1] \geq \Pr[b' = 1 | b = 0]$; sinon $\mathcal{A}'$ renvoie $1 - b'$ au lieu de b'. Ceci implique $\Pr[Win] \geq 1/2 + \varepsilon/2$. □

Le facteur $1/2$ qui sépare les deux fait que certains auteurs utilisent $2|\Pr[Win] - 1/2|$ au lieu de $|\Pr[Win] - 1/2|$. Mais dans le cadre de ce cours, la définition est équivalente puisque $2\text{negl}(n) = \text{negl}(n)$.

Indistinguabilité Statistique.

Définition 10 (Distance statistique)**.** Soit X, Y deux variables aléatoires sur un ensemble dénombrable A. La distance statistique entre X et Y est définie par :

$$\Delta(X,Y) = \frac{1}{2}\sum_{a\in A} |\Pr[X=a] - \Pr[Y=a]| \ . \tag{13}$$

On peut vérifier qu'il s'agit effectivement d'une distance :

- $\Delta(X,Y) \geq 0$ et vaut 0 ssi X et Y ont la même distribution ;
- $\Delta(X,Y) = \Delta(Y,X)$
- $\Delta(X,Z) \leq \Delta(X,Y) + \Delta(Y,Z)$

Définition 11. Deux distributions D_0, D_1 sont statistiquement indistinguables s'il existe une fonction négligeable negl telle que :

$$\Delta(D_0, D_1) \leq \text{negl}(n)$$

C'est une propriété plus forte que l'indistinguabilité calculatoire.

2.4 Sécurité du Chiffrement à Clé Publique

Souvenons-nous qu'un schéma a une sécurité parfaite si un adversaire avec une puissance de calcul infinie ne peut rien apprendre du message clair étant donné le chiffré. Nous avons vu que le One-Time Pad offrait une telle sécurité. Mais ce n'est pas utilisable dans le contexte du chiffrement à clé publique, puisque la clé publique a vocation à être utilisée plusieurs fois ; et de plus, une clé trop longue est inenvisageable.

La **sécurité sémantique** a la même définition que la sécurité parfaite, mais où l'adversaire est limité à une puissance de calcul polynomiale. C'est-à-dire que pour toute distribution de probabilité sur l'espace des messages, tout ce que l'adversaire peut calculer en temps polynomial sur le message clair étant donné le chiffré, iel peut également le calculer sans le chiffrer.

Malheureusement, cette notion de sécurité n'est pas facile à prouver en pratique. On utilise à la place la notion d'**indistinguabilité des chiffrés**, qui s'avère équivalente à la sécurité sémantique (bien que ce ne soit pas évident).

L'indistinguabilité formalise le fait qu'un attaquant qui a choisi deux messages, et à qui on donne le chiffré de l'un d'entre eux, n'est pas capable de dire auquel des deux messages correspond ce chiffré. Il existe différents « niveaux » d'indistinguabilité en fonction de l'accès que l'attaquant a au schéma cryptographique : attaques à **clair choisi** (CPA, *chosen-plaintext attacks*) et à **chiffré choisi** (CCA, *chosen-ciphertext attack*).

Définition 12 (Sécurité CPA)**.** Le jeu de sécurité pour la sécurité CPA est défini comme suit.

1. **Initialisation :** Le Challenger $\mathcal{C}$ choisit un bit $b \hookleftarrow U(0,1)$ et des clés $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$. Il envoie pk à l'adversaire $\mathcal{A}$.
2. **Find stage :** $\mathcal{A}$ choisit une paire de messages m_0, m_1 et l'envoie à $\mathcal{C}$, qui renvoie $c^* = \mathsf{Enc}(\mathsf{pk}, m_b)$ (le « chiffré challenge »)
3. **Guess stage :** $\mathcal{A}$ calcule un bit b' et le renvoie ; il gagne le jeu si $b = b'$.

L'avantage de l'adversaire est défini par :

$$\text{Adv}^{CPA}(\mathcal{A}) = \left|\Pr[\mathcal{A} \text{ Gagne}] - \frac{1}{2}\right| \ . \tag{14}$$

Si l'avantage de tout adversaire PPT est negl(n), alors le chiffrement est IND-CPA (*IND-CPA secure*).

Quelques remarques :

- Il est plus courant de définir l'avantage par $2|\Pr[\mathcal{A}Wins] - \frac{1}{2}|$ mais ça ne change rien à la définition puisque 2negl = negl.
- Durant le jeu, l'adversaire, qui dispose de la clé publique, peut chiffrer n'importe quel message de son choix (d'où la notion de « clair choisi »).
- Le chiffrement doit être nécessairement probabiliste, sinon il existe une attaque triviale en chiffrant soi-même les messages m_0 et m_1.
- Cette notion implique aussi qu'une clé peut être utilisée plusieurs fois.

Sécurité CCA. La sécurité CCA autorise l'adversaire à effectuer non seulement des chiffrements, mais aussi des déchiffrements. Comme il ne dispose pas de la clé secrète, il doit faire des **requêtes** de déchiffrement au challenger.

Toutefois, s'il était autorisé à déchiffrer n'importe quel message, cette définition n'aurait plus de sens : il pourrait déchiffrer le chiffré challenge $c^* = \mathsf{Enc}(\mathsf{pk}, m_b)$. On interdit donc spécifiquement de déchiffrer celui-ci. On parle alors de sécurité contre des attaques à **chiffré choisi** (CCA = *chosen ciphertext atack*).

Définition 13 (Sécurité CCA)**.** Le jeu de sécurité pour la sécurité CCA est défini comme suit.

1. **Initialisation** Le Challenger $\mathcal{C}$ choisit un bit $b \hookleftarrow U(0,1)$ et des clés $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^n)$. Il envoie pk à l'adversaire $\mathcal{A}$.
2. **Find stage** $\mathcal{A}$ choisit une paire de messages m_0, m_1 et l'envoie à $\mathcal{C}$, qui renvoie $c^* = \mathsf{Enc}(\mathsf{pk}, m_b)$
3. **Guess stage** $\mathcal{A}$ calcule un bit b' et le renvoie ; il gagne le jeu si $b = b'$.

De plus, au cours du jeu, l'adversaire peut effectuer une requête de déchiffrement. Il choisit un ciphertext c et l'envoie à $\mathcal{C}$. Si $c \neq c^*$, $\mathcal{C}$ renvoie $\mathsf{Dec}(\mathsf{sk}, c)$, sinon $\perp$. Dans le jeu IND-CCA1 (non adaptatif), l'attaquant peut faire des requêtes de déchiffrement uniquement dans la "find stage" (avant de connaître c^*). Dans le jeu IND-CCA2 (adaptatif), l'attaquant peut faire des requêtes à n'importe quel moment.

L'avantage de l'adversaire est défini par :

$$\mathrm{Adv}^{CCA}(\mathcal{A}) = \left|\Pr[\mathcal{A}Wins] - \frac{1}{2}\right| \quad . \tag{15}$$

Si l'avantage de tout adversaire PPT est negl(n), alors le chiffrement est IND-CCA(1,2) (IND-CCA secure).

Résumé

- En anglais « symétrique » se dit *symmetric*, « asymétrique » se dit *asymmetric*. Pensez au nombre de *m*.
- Indistinguabilité : étant donné des échantillons d'une distribution, l'adversaire ne sait pas de laquelle il s'agit.
- IND-CPA : l'adversaire ne sait pas quel message a été chiffré (même s'il a choisi le message).
- IND-CCA : comme IND-CPA, mais avec accès à un oracle de déchiffrement.

3 Construire un Chiffrement à Clé Publique : RSA

Un chiffrement à clé publique nécessite :

- Un problème « difficile à inverser » : qui permette de construire une clé publique à partir d'une clé secrète, mais pas de déduire la clé secrète de la clé publique ;
- Une génération de clés efficace (en temps polynomial).

Le schéma de chiffrement à clé publique le plus célèbre, et encore l'un des plus utilisés à ce jour, est le chiffrement RSA. Dans RSA la clé secrète est (entre autres) un couple de deux nombres premiers distincts (P, Q) et la clé publique est (entre autres) le produit des nombres P et Q. Faire le produit de deux nombres est « facile » (polynomial). Retrouver P et Q à partir de N est difficile (non polynomial). Enfin, générer des nombres premiers est également facile.

3.1 Génération des Nombres Premiers

Le théorème des nombres premiers (*prime number theorem*) nous apprend que les nombres premiers sont plutôt nombreux : en sélectionnant un entier de n bits au hasard on obtient un nombre premier avec probabilité $\mathcal{O}(1/n)$.

Théorème 2 (Théorème des nombres premiers)**.** *Il existe une constante c telle que pour tout $n > 1$, il y a au moins $c2^{n-1}/n$ nombres premiers de n bits.*

Test de primalité de Fermat. Le test de Fermat consiste à remarquer que pour un nombre premier on a toujours : $a^{p-1} = 1 \mod p$ d'après le petit théorème de Fermat, mais que pour p non premier on a $a^{p-1} \neq 1 \mod p$ avec une probabilité constante.

Algorithm 1 Fermat primality test.

1: **for** $j = 0$ to $k - 1$ **do**
2: Pick $a \in [2, ...p - 2]$
3: $b \leftarrow a^{p-1} \mod p$
4: **If** $b \neq 1$ return Composite
5: **end for**
6: **Return** "probable prime"

Malheureusement le test échoue sur certains nombres.

Test de Primalité de Miller-Rabin. Le test de Miller-Rabin est une modification du test de Fermat, qui s'appuie sur les propriétés suivantes, plus fortes.

Théorème 3. *Soit $p > 2$ un nombre premier, soient s et d les deux entiers (d impair) tels que $p = 2^s d + 1$. Alors pour tout a n'étant pas divisible pas p :*

$$a^d = 1 \pmod p \text{ ou } \exists r < s, a^{2^r d} = -1 \pmod p \ . \tag{16}$$

Démonstration. D'après le petit théorème de Fermat : $a^{p-1} = (a^d)^{2^s} = 1 \pmod p$. En prenant de façon répétée les racines carrées de a^{p-1}, on obtient soit toujours 1 modulo p jusqu'à a^d, où on s'arrête pour un certain r tel que $a^{d2^r} = -1 \pmod p$. □

Par contraposée, si $a^d \neq 1 \pmod p$ et $\forall r < s, a^{2^r d} \neq -1 \pmod p$, alors p est composé, et a est appelé un témoin de Miller pour le fait que p est composé.

Si l'équation 16 est vérifiée pour a, on dit que p est fortement probablement premier en base a. On a la propriété suivante.

Proposition 1. *Pour un nombre impair composé p, 3/4 au moins des entiers a, $1 < a < p$ sont des témoins de Miller pour p.*

Soit p un nombre entier, écrivons p sous la forme $p = 2^s d + 1$ où s est un entier et d est impair.

Le test de Miller-Rabin consiste donc à choisir a au hasard, vérifier si c'est un témoin de Miller, et répéter le processus k fois.

Algorithm 2 Test de primalité de Miller-Rabin.

Write $p - 1 = 2^s d$ with d odd
for $j = 0$ to $k - 1$ **do**
    Pick $a \in [2, ...p - 2]$ u.a.r.
    $b \leftarrow a^d \mod p$
    If $b = 1$ of $b = p - 1$ **Return** "composite"
    $i \leftarrow 1$
    for $i = 1$ to $s - 1$ **do**
        $b \leftarrow b^2 \mod p$
        If $b = p - 1$ return "composite"
    end for
end for
Return "probable prime"

Il n'y a pas de faux négatif et la probabilité d'un faux positif est $\leq 1/4^k$. Pour terminer, il suffit d'effectuer ce test sur des nombres pris au hasard.

3.2 Factorisation

La factorisation (des entiers RSA) est le problème suivant.

Problème 1 (Factorisation)**.** *On tire deux nombres premiers P, Q de n bits au hasard.*

- *Entrée : $N = PQ$*
- *Sortie : P, Q*

Ce n'est **pas** exactement le problème difficile sur lequel repose RSA, qui est « l'hypothèse RSA » que nous définirons plus tard. En fait, on ne sait pas dire si la factorisation est équivalente au problème RSA, même si la meilleure voie d'attaque que l'on connaît passe par la factorisation.

La méthode naïve de factorisation (énumérer les premiers et chercher une divisibilité) est très loin d'être la meilleure. Aujourd'hui, le meilleur algorithme connu à ce jour est le **General Number Field Sieve** [Pom96], dont le temps de calcul est subexponentiel :

$$\exp\left[\left((64/9)^{1/3} + o(1)\right)(\log n)^{1/3}(\log\log n)^{2/3}\right] \quad ,$$

où n est la taille en bits du nombre. C'est **beaucoup moins** qu'une complexité exponentielle en n. Les records de calculs utilisent toujours des variantes du NFS [Bou+20].

3.3 RSA

Le chiffrement RSA, encore le plus utilisé à ce jour, a été introduit en 1977 par Rivest, Shamir et Adleman [RSA78]. Dans la suite ϕ est l'indicatrice d'Euler, qui vaut $(P-1)(Q-1)$ pour $N = PQ$, et les calculs sont faits dans $\mathbb{Z}_N^*$ (modulo N).

“Textbook” RSA PKE

KeyGen :

- Choisir P, Q premiers, $N = PQ$, e, d tels que $ed = 1 \pmod{\phi(N)}$.
- $\mathsf{sk} = d, \mathsf{pk} = (N, e)$

Enc ($\mathbf{m} \in \mathbb{Z}_N^*$) :

- $\mathbf{c} = \mathbf{m}^e$

Dec :

- $\mathbf{m} = \mathbf{c}^{\mathbf{d}}$.

La correction du schéma provient de :

$$(m^e)^d = m^{ed} = m \pmod{N} \ . \tag{17}$$

Pour trouver e et d, on commence par choisir e qui est premier avec $\phi(N)$, puis on calcule son inverse modulo $\phi(N)$.

Sécurité du Chiffrement RSA. La sécurité est définie par le jeu suivant entre $\mathcal{C}$ et $\mathcal{A}$.

- $\mathcal{C}$ génère (N, e, d) et $y = U(\mathbb{Z}_N^*)$ et renvoie (N, e, y)
- $\mathcal{A}$ renvoie x
- $\mathcal{C}$ renvoie “Win” si $x^e = y \mod N$

L’avantage de $\mathcal{A}$ est la probabilité de gagner le jeu. L’**hypothèse RSA** est que ce problème est difficile. On sait que :

Lemme 4. *S’il existe un adversaire PPT pour le problème de factorisation, il existe un adversaire PPT pour le problème RSA.*

Démonstration. De P et Q, on peut calculer $\phi(N)$ et ainsi recalculer la clé secrète d. □

En d’autres termes la factorisation est **plus** difficile que le problème RSA. On ne connaît pas de réduction dans l’autre sens, bien que les meilleurs attaques connues passent par la factorisation de N. Notez qu’il existe aussi d’autres attaques si les paramètres sont mal choisis ; certaines seront vues en TD.

3.4 Padded RSA

Est-ce que ce chiffrement est IND-CPA ? C’est mal parti : il est déterministe. Pour y remédier on ajoute un *padding* aléatoire au message de départ. On change de domaine pour le message : on prend $\{0,1\}^\ell$ où $\ell < \log_2 N$.

Padded RSA PKE

KeyGen :

- Choisir P, Q premiers, $N = PQ$, e, d tels que $ed = 1 \pmod{\phi(N)}$.
- $\mathsf{sk} = d, \mathsf{pk} = (N, e)$

Enc $m \in \{0,1\}^\ell$

- Choisir $r \hookleftarrow U(\{0,1\}^{\log_2 N - \ell})$
- Calculer $m' \in \mathbb{Z}_N$ dont la représentation binaire est $(r \| m)$ (la concaténation de r et m).
- Renvoyer $c = (m')^e$.

Dec :

- Renvoyer les ℓ bits de poids faible de $m = c^d \mod N$.

La sécurité dépend de ℓ. Si ℓ est trop grand, une attaque par force brute permettra de retrouver le message. Si $\ell = \mathcal{O}(\log N)$, on peut prouver la sécurité sous l'hypothèse RSA.

En pratique N a 2048 bits au moins. Les premiers P, Q ont 1024 bits. L'exposant e est de poids de Hamming faible (i.e., il y a peu de 1 dans son écriture binaire) afin de rendre le chiffrement rapide (il est donc plus rapide que le déchiffrement, car d n'est a priori pas de poids faible).

Résumé

- Générer des nombres pseudo-premiers est efficace à l'aide du test de Miller-Rabin.
- RSA repose sur le « problème RSA » qui, à ce que nous savons, n'est pas équivalent à la factorisation.
- La factorisation est un problème difficile, mais pas exponentiel.
- RSA "textbook" ne **doit pas** être utilisé : il faut l'associer à un remplissage. De plus, de nombreux choix de paramètres sont à éviter.

Références

[Bou+20] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé et Paul Zimmermann. "Comparing the Difficulty of Factorization and Discrete Logarithm : A 240-Digit Experiment". In : *CRYPTO (2)*. T. 12171. Lecture Notes in Computer Science. Springer, 2020, p. 62-91.

[Pom96] Carl Pomerance. "A tale of two sieves". In : *Notices of the American Mathematical Society* 43.12 (1996), p. 1473-1485.

[RSA78] Ronald L. Rivest, Adi Shamir et Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In : *Commun. ACM* 21.2 (1978), p. 120-126. doi : 10.1145/359340.359342. url : https://doi.org/10.1145/359340.359342.

4 DLP, DH et ElGamal

Un autre problème très utilisé en cryptographie à clé publique est le problème du logarithme discret (dans un groupe bien choisi).

Soit un groupe $\mathbf{G}$ d'ordre donné et un générateur g de $\mathbf{G}$. Le problème du logarithme discret (discrete logarithm, ou DL) est, sur l'entrée g^a où a a été choisi uniformément au hasard, de retrouver a.

Problème 2 (DL)**.** *On tire a uniformément au hasard.*

- *Entrée : g^a*
- *Sortie : a*

Attention, la difficulté du problème dépend fortement du groupe choisi. Par exemple, si l'on prend le sous-groupe de $(\mathbb{Z}_N, +)$ engendré par un nombre k premier avec N, le problème revient à trouver a étant donné un multiple $x = ak \pmod N$ de k. Il suffit de multiplier par k^{-1}.

Historiquement le choix s'est donc porté sur un groupe multiplicatif $(\mathbb{Z}_p^*, \cdot)$ où p est premier. C'est un groupe cyclique, facile à générer. Cependant l'ordre de ce groupe n'est pas premier (plus généralement un groupe d'ordre friable est problématique) et le problème décisionnel du DLP n'est pas difficile.

Un bon choix est donc un sous-groupe bien choisi de $(\mathbb{Z}_p^*, \cdot)$, par exemple le groupe des résidus quadratiques (les nombres qui sont des carrés modulo p). De nos jours, on utilise plutôt les groupes de points sur des courbes elliptiques définies sur des corps finis.

4.1 Résoudre le DLP

Dans un groupe quelconque d'ordre p, la meilleure complexité asymptotique connue pour résoudre le DLP est en $\mathcal{O}\big(\sqrt{p}\big)$ (si p a n bits, c'est $2^{n/2}$, qui est une complexité exponentielle). Un algorithme simple consiste à faire appel au paradoxe des anniversaires.

Lemme 5 (Paradoxe des anniversaires)**.** *Soit $N \in \mathbb{N}$ et $y_1, \ldots, y_\ell$ choisis uniformément au hasard dans un ensemble de taille N. Alors la probabilité qu'il existe deux indices distincts i, j tels que $y_i = y_j$ est au plus $\ell^2/2N$. Si $\ell \leq \sqrt{2N}$, elle est au moins $\frac{\ell(\ell-1)}{4N}$.*

L'intuition derrière le paradoxe des anniversaires (qui n'est pas un paradoxe !) est que lorsque l'on prend ℓ éléments aléatoires, ils forment environ ℓ^2 paires. Pour chacune de ces paires, la probabilité de collision est d'environ $\frac{1}{N}$, donc on peut s'attendre à une collision avec une probabilité de $\frac{\ell^2}{N}$. **Cependant**, les paires ne sont pas indépendantes, donc une preuve formelle est nécessaire.

Démonstration. Soit $Coll_{i,j}$ l'événement : $y_i = y_j$ et $Coll$ l'événement d'une collision. Nous avons : $\forall i \neq j, \Pr[Coll_{i,j}] = 1/N$, et $Coll = \bigcup_{i \neq j} Coll_{i,j}$, où l'union est faite sur toutes les paires disjointes (i, j). En utilisant l'inégalité de Boole :

$$\Pr[Coll] \leq \sum_{i \neq j} \Pr[Coll_{i,j}] = \binom{\ell}{2} \frac{1}{N} \leq \frac{\ell^2}{2N} \ . \tag{18}$$

Soit $NoColl_i$ l'événement qu'il n'y ait pas de collision parmi $y_1, \ldots, y_i$. En particulier, $NoColl_i$ implique $NoColl_{i-1}$. Nous avons :

$$\Pr[NoColl_\ell] = \Pr[NoColl_1] \cdot \Pr[NoColl_2 | NoColl_1] \cdots \Pr[NoColl_\ell | NoColl_{\ell-1}] \ . \tag{19}$$

De plus, $\Pr[NoColl_1] = 1$ (y_1 ne peut pas entrer en collision avec lui-même). En outre, si $NoColl_i$ est vrai, alors $NoColl_{i+1}$ ne peut être faux que si le nouvel élément y_{i+1} forme une

paire en collision avec l'un des $y_1, \ldots, y_i$. Cela ne peut se produire qu'avec une probabilité de $\frac{i}{N}$. Ainsi :

$$\Pr\left[NoColl_{i+1} | NoColl_i\right] = 1 - i/N \tag{20}$$

$$\Pr\left[NoColl_\ell\right] = \prod_{i=1}^{\ell-1}(1 - i/N) \tag{21}$$

Nous allons majorer cette expression en utilisant l'identité : $1 - i/N \leq e^{-i/N}$:

$$\Pr\left[NoColl_\ell\right] \leq e^{-\sum_{i=1}^{\ell-1} i/N} = e^{-\ell(\ell-1)/2N} \quad . \tag{22}$$

Nous utiliserons également l'identité $1 - x/2 \geq e^{-x}$ pour $x < 1$:

$$\Pr\left[Coll\right] = 1 - \Pr\left[NoColl_\ell\right] \geq 1 - e^{-\ell(\ell-1)/2N} \geq \frac{\ell(\ell-1)}{4N} \quad . \tag{23}$$

□

C'est un résultat **extrêmement important** pour la cryptographie, tant symétrique qu'asymétrique, aussi bien pour la sécurité prouvable que pour la cryptanalyse. Lorsque $\ell \leq \sqrt{N}$, une collision se produit avec une probabilité $\Theta(\ell^2/N)$. Pour $\ell = \sqrt{N}$, cette probabilité devient constante. Lorsque $\ell > \sqrt{N}$, c'est la probabilité de **ne pas trouver** de collision qui devient faible.

On appelle cela le paradoxe des anniversaires car dans une pièce de ℓ personnes, la probabilité que deux d'entre elles partagent le même anniversaire est :

$$1 - (1 - 1/365)(1 - 2/365) \cdots (1 - (\ell - 1)/365) \tag{24}$$

ce qui est supérieur à $1/2$ lorsque $\ell = 23$.

Résolution du DLP avec les anniversaires. Étant donné $h = g^a$ et g, nous voulons trouver a. Nous pouvons le faire en complexité $\mathcal{O}\big(\sqrt{p}\big)$ en prenant environ $\sqrt{p}$ entiers $i \in \mathbb{Z}_p$ et en testant si $h^i = h^j$ pour une paire de i, j distincts. Si cela se produit :

$$a \cdot i = a \cdot j \mod p \implies a = (i - j)^{-1} \mod p \tag{25}$$

Dans $\mathbb{Z}_p^*$, le meilleur algorithme pour résoudre le DLP est bien meilleur. Comme la factorisation, il s'agit d'un algorithme de crible (Generalized Number Field Sieve) en complexité sous-exponentielle $\exp(\mathcal{O}\big((\log p)^{1/3}(\log\log p)^{2/3}\big))$.

L'essor des courbes elliptiques en cryptographie à clé publique s'explique notamment par le fait que les algorithmes de résolution du DLP y sont bien plus inefficaces, et on peut donc obtenir un niveau de sécurité élevé avec des clés plus petites. (Vous pouvez par exemple le constater en générant des clés SSH pour RSA ou pour ECC).

4.2 Échange de Clés Diffie-Hellman

L'échange de clés DH est un protocole dans lequel deux parties Alice et Bob s'accordent sur une clé secrète partagée. Le protocole s'exécute ainsi.

- Paramètres : un groupe $\mathbf{G}$ et un générateur g d'ordre n
- Alice choisit un entier $\mathbf{a} \hookleftarrow U([1; n-1])$, calcule $\mathbf{A} = g^{\mathbf{a}}$ et l'envoie à Bob
- Bob choisit un entier $\mathbf{b} \hookleftarrow U([1; n-1])$, calcule $\mathbf{B} = g^{\mathbf{b}}$ et l'envoie à Alice
- Alice calcule $\mathbf{B}^{\mathbf{a}}$
- Bob calcule $\mathbf{A}^{\mathbf{b}}$

La valeur $\mathbf{B}^{\mathbf{a}} = (g^{\mathbf{b}})^{\mathbf{a}} = g^{\mathbf{ab}} = \mathbf{A}^{\mathbf{b}}$ est la clé partagée d'Alice et Bob.
La sécurité de l'échange de clés DH repose sur les problèmes suivants.

Problème 3 (Problème de Diffie-Hellman décisionnel (DDH = *decisional Diffie-Hellman*))**.** *Distinguer entre* (g^a, g^b, g^{ab}) *et* (g^a, g^b, g^c) *pour* a, b, c *uniformes et indépendants.*

Problème 4 (Problème de Diffie-Hellman calculatoire (CDH = *computational Diffie-Hellman*))**.** *On tire* $a, b \hookleftarrow U([1; n-1])$.

- *Entrée :* g^a, g^b
- *Sortie :* g^{ab}

Notons que :

- CDH est plus difficile que DDH (si l'on peut résoudre CDH en temps polynomial, alors DDH aussi)
- DL est plus difficile que CDH

Il existe des groupes dans lesquels CDH est difficile et DDH est facile.

La difficulté du problème DDH peut aussi être définie par deux jeux $RAND$ et DDH entre $\mathcal{C}$ et $\mathcal{A}$:

- $\mathcal{C}$ génère $\mathsf{KeyGen}(1^n) = (\mathbf{G}, q, g)$
- $\mathcal{C}$ choisit $x, y \leftarrow U(\mathbb{Z}_q)$
- Dans le cas RAND $(b = 0)$: $z \leftarrow U(\mathbb{Z}_q)$; dans le cas DDH $(b = 1)$: $z = xy$
- $\mathcal{C}$ envoie (g, g^x, g^y, g^z) à $\mathcal{A}$
- $\mathcal{A}$ renvoie un bit b'

L'avantage de $\mathcal{A}$ est :

$$\mathrm{Adv}(\mathcal{A}) = \left| \Pr\left[\mathcal{A} \xrightarrow{RAND} 1\right] - \Pr\left[\mathcal{A} \xrightarrow{DDH} 1\right] \right| \ . \tag{26}$$

Le problème DDH est difficile dans le groupe $\mathbf{G}$ si pour tout adversaire PPT $\mathcal{A}$ il existe une fonction négligeable negl telle que $\mathrm{Adv}(\mathcal{A}) \leq \mathrm{negl}$.

4.3 Chiffrement ElGamal

Dans la suite nous travaillons dans un groupe $\mathbf{G}$ où le problème DDH est difficile. Le chiffrement ElGamal a été introduit en 1984 par ElGamal. Il est par exemple utilisé dans GPG (Gnu Privacy Gard) qui permet de transmettre des e-mails chiffrés et signés.

Chiffrement ElGamal

Paramètres publics : $(\mathbf{G}, q, g)$ (q est l'ordre de $\mathbf{G}$, g un générateur)
KeyGen :

- Tirer $x \leftarrow U(\mathbb{Z}_q)$
- $\mathsf{sk}, \mathsf{pk} = \mathbf{x}, g^{\mathbf{x}}$

Enc $\mathbf{m} \in \mathbf{G}$

- Tirer $\mathbf{y} \leftarrow U(\mathbb{Z}_q)$
- Renvoyer $\mathbf{c_1}, \mathbf{c_2} := (g^{\mathbf{y}}, h^{\mathbf{y}} \cdot \mathbf{m})$

Dec $\mathbf{c} = (\mathbf{c_1}, \mathbf{c_2})$

- Return $\mathbf{m} = \mathbf{c_2}(\mathbf{c_1}^{-\mathbf{x}})$

Correction.

$$c_2(c_1^{-x}) = h^y m g^{-xy} = g^{xy} m g^{-xy} = m \ . \tag{27}$$

Théorème 4. *Si DDH est difficile dans* $\mathbf{G}$*, alors ElGamal est IND-CPA.*

Nous faisons cette preuve par **réduction** : nous montrons que s'il existe un adversaire (c'est-à-dire un algorithme) efficace pour attaquer ElGamal avec un avantage non négligeable, alors il existe également un adversaire pour attaquer DDH avec un avantage non négligeable. Ainsi on montre bien que DDH est un problème plus facile que ElGamal, ou en d'autres termes, qu'on peut **réduire** ElGamal à DDH (toute cette terminologie est importante).

En pratique, on suppose donné un adversaire pour le jeu IND-CPA sur ElGamal. On utilise cet adversaire en boîte noire, comme un **oracle**, pour écrire un algorithme qui gagne le jeu DDH.

Démonstration. Considérons d'abord le jeu IND-CPA pour ElGamal :

- Initialisation : $\mathcal{C}$ appelle $\mathsf{KeyGen} = (x, g^x)$, choisit $b \hookleftarrow U(0,1)$ et envoie g^x à $\mathcal{A}$
- $\mathcal{A}$ choisit m_0, m_1 et les envoie à $\mathcal{C}$
- $\mathcal{C}$ calcule $c_1, c_2 = \mathsf{Enc}(\mathsf{pk}, m_b)$ et renvoie (c_1, c_2) à $\mathcal{A}$
- $\mathcal{A}$ calcule un bit b', si $b' = b$ Win

Nous allons montrer que si DDH est difficile, il existe une fonction negl telle que $\mathrm{Adv}^{CPA}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ Win}] - 1/2| \leq \mathrm{negl}(n)$.

Pour ce faire, nous allons utiliser $\mathcal{A}$ pour définir un adversaire $\mathcal{B}$ contre DDH. Plaçons-nous donc désormais dans les jeux RAND et DDH.

- $\mathcal{C}$ génère $\mathsf{KeyGen}(1^n) = (\mathbf{G}, q, g)$
- $\mathcal{C}$ choisit $x, y \hookleftarrow U(\mathbb{Z}_q)$
- Dans le cas RAND ($b = 0$) : $z \hookleftarrow U(\mathbb{Z}_q)$; dans le cas DDH ($b = 1$) : $z = xy$
- $\mathcal{C}$ envoie (g, g^x, g^y, g^z) à $\mathcal{B}$
- $\mathcal{B}$ envoie g, g^x à $\mathcal{A}$
- $\mathcal{A}$ choisit m_0, m_1 et les envoie à $\mathcal{B}$
- $\mathcal{B}$ choisit b', calcule $g^y, g^z \cdot m_{b'}$ et l'envoie à $\mathcal{A}$
- $\mathcal{A}$ renvoie b''
- Si $b' = b''$ ($\mathcal{A}$ a réussi), $\mathcal{B}$ renvoie 1, sinon 0

On peut le représenter ainsi :

$\mathcal{C}$		$\mathcal{B}$		$\mathcal{A}$
RAND : $z \hookleftarrow U(\mathbb{Z}_q)$				
DDH : $z = xy$	$\xrightarrow{g, g^x, g^y, g^z}$		$\xrightarrow{g, g^x}$	m_0, m_1
		Choose b	$\xleftarrow{m_0, m_1}$	
		$c_1, c_2 := (g^y, g^z \cdot m_b)$	$\xrightarrow{c_1, c_2}$	Compute b'
		If $b = b'$ output 1 else 0	$\xleftarrow{b'}$	

Concentrons-nous sur $\mathcal{B}$.

- Dans le cas RAND ($b = 0$) : z est uniforme et indépendant du reste, donc c_2 également. $\mathcal{A}$ ne peut pas distinguer entre les deux chiffrés. La probabilité que $\mathcal{B}$ renvoie 1 est donc exactement $1/2$.
- Dans le cas DDH : $z = xy$ et le chiffré est un chiffré ElGamal valide. La probabilité que $\mathcal{B}$ renvoie 1 est la probabilité que $\mathcal{A}$ réussisse dans le jeu IND-CPA :

$$\Pr\left[\mathcal{B} \xrightarrow{DDH} 1\right] = \Pr[\mathcal{A} \text{ Win}] \tag{28}$$

Par hypothèse, DDH est difficile, donc il existe une fonction négligeable telle que :

$$\mathrm{Adv}^{CPA}(\mathcal{A}) = |1/2 - \Pr[\mathcal{A}Win]| = |\Pr\left[\mathcal{B} \xrightarrow{RAND} 1\right] - \Pr\left[\mathcal{B} \xrightarrow{DDH} 1\right]| \leq \mathrm{negl} \ . \tag{29}$$

C'est le cas pour tout adversaire $\mathcal{A}$, ce qui prouve que ElGamal est IND-CPA. □

4.4 Transformée de Fujisaki-Okamoto

La transformation Fujisaki-Okamoto (FO dans ce qui suit) [FO13] transforme un PKE IND-CPA en un PKE IND-CCA2. Notez qu'il existe plusieurs variantes de la transformation et nous n'en présenterons qu'une ici.

Spécification. Considérons un schéma de chiffrement à clé publique qui est sécurisé contre les attaques CPA. Par définition, le schéma n'est pas déterministe, et la fonction de chiffrement peut être écrite sous la forme : $E(\mathbf{m}, \mathbf{r})$ où $\mathbf{m}$ est le message secret et $\mathbf{r}$ l'aléa. Soit D la fonction de déchiffrement. Soit H une fonction de hachage cryptographiquement sécurisée (pour les raisons expliquées dans le paragraphe suivant).

Définissons :

$$E'(\mathbf{m}, \mathbf{r}) = E(\mathbf{m}|\mathbf{r}, H(\mathbf{m}|\mathbf{r})) \tag{30}$$

et modifions le déchiffrement en calculant d'abord $\mathbf{m'} = D(\mathbf{c})$ **et** en vérifiant que $\mathbf{c} = E(\mathbf{m'}, H(\mathbf{m'}))$.

Ce n'est que si cette équation est vérifiée que le déchiffrement renverra $\mathbf{m'} = \mathbf{m}|\mathbf{r}$ (et ainsi nous avons récupéré $\mathbf{m}$). Si l'équation n'est pas vérifiée, $\perp$ est retourné.

Sécurité. Il existe une preuve de sécurité qui montre que **dans le modèle d'oracle aléatoire**, la sécurité IND-CPA de (E, D) implique la sécurité IND-CCA2 de (E', D'). Le modèle d'oracle aléatoire suppose que la fonction de hachage H ne se comporte pas différemment d'une fonction choisie aléatoirement parmi toutes les fonctions possibles (le soi-disant "oracle aléatoire"). C'est évidemment une hypothèse forte qui n'a de sens que si H est une bonne fonction de hachage cryptographique. En particulier, elle devrait être sécurisée contre les attaques par collision et par préimage. Les choix typiques sont les familles SHA-2 et SHA-3.

Résumé

- La difficulté du DL (resp. de DDH et CDH) dépend du groupe choisi.
- ElGamal est IND-CPA, **mais pas IND-CCA** (voir TDs). Il ne doit pas être utilisé tel quel.
- La transformée de FO permet de transformer génériquement un chiffrement à clé publique IND-CPA en un chiffrement IND-CCA2. Des variantes de FO sont couramment utilisées dans la pratique.

5 Chiffrement Basé sur LWE

Le problème Learning with Errors (LWE) a été introduit par Regev en 2005. Il s'agit aujourd'hui de la pierre angulaire de la **cryptographie basée sur les réseaux euclidiens**. Ironiquement, il n'y aura aucun réseau euclidien dans cette section : c'est d'ailleurs l'un des intérêts de LWE. Il est beaucoup plus facile à utiliser que les précédents cryptosystèmes basés sur les réseaux euclidiens, bien que sa sécurité puisse être réduite aux mêmes problèmes difficiles.

L'intérêt de LWE est d'être résistant à un adversaire quantique : on parle de **cryptographie post-quantique**.

Dans cette section on utilise **le texte gras** pour noter des objets multi-dimensionnels tels que des vecteurs (ligne ou colonne) et matrices.

5.1 Bad-LWE

Ouvrons cette section avec un mauvais schéma de chiffrement à clé publique, que nous appellerons « Bad-LWE » . Le cryptosystème est mauvais, mais la notation est bonne, donc gardez-la à l'esprit. Dans Bad-LWE, la clé privée est un vecteur $\mathbf{s} \in \mathbb{Z}_q^n$ sélectionné (uniformément) au hasard. Pour la clé publique, nous sélectionnons une matrice $\ell \times n$ $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$ (uniformément) au hasard.

Nous publions $\mathbf{A}$ et $\mathbf{b} := \mathbf{A}\mathbf{s} \in \mathbb{Z}_q^\ell$. Ici, $\mathbf{s}$ est considéré comme un vecteur colonne et $\mathbf{b}$ également. Typiquement, les lignes de $\mathbf{A}$ peuvent être notées $\mathbf{a_i}$ et les éléments de $\mathbf{b}$ comme $\mathbf{b_i} := \mathbf{a_i} \cdot \mathbf{s}$.

De plus, pour tout vecteur $\mathbf{r} \in \{0,1\}^\ell$, considéré cette fois comme une ligne, nous pouvons faire l'opération $\mathbf{rA}$; le résultat est une combinaison linéaire des lignes de $\mathbf{A}$. Enfin, $\mathbf{rAs} = \mathbf{r} \cdot \mathbf{b}$ où $\cdot$ est un produit scalaire de vecteurs ℓ-dimensionnels de $\mathbb{Z}_q$.

Bad-LWE

KeyGen :

- Clé privée : $\mathbf{s} \in \mathbb{Z}_q^n$ choisi aléatoirement
- Clé publique : $(\mathbf{A}, \mathbf{b} := \mathbf{A}\mathbf{s})$ où $\mathbf{A}$ est une matrice aléatoire de ℓ lignes et n colonnes : $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$

Enc $\mathbf{m} \in \{0,1\}$:

- Choisir un vecteur aléatoire $\mathbf{r} \in \{0,1\}^\ell$
- Retourner $\mathbf{c_1}, \mathbf{c_2} := \mathbf{rA}, (\mathbf{m} + \mathbf{r} \cdot \mathbf{b})$

Dec $\mathbf{c} = (\mathbf{c_1}, \mathbf{c_2}) \in \mathbb{Z}_q^{n+1}$:

- Retourner $\mathbf{m} = \mathbf{c_2} - \mathbf{c_1} \cdot \mathbf{s}$

Commençons par remarquer que le déchiffrement dans Bad-LWE fonctionne. Cela est simplement dû au fait que les produits matriciels sont associatifs :

$$\begin{aligned} \mathbf{c_2} - \mathbf{c_1} \cdot \mathbf{s} &= (\mathbf{m} + \mathbf{r} \cdot \mathbf{b}) - (\mathbf{rA}) \cdot \mathbf{s} \\ &= \mathbf{m} + (\mathbf{r} \cdot \mathbf{b}) - \mathbf{r}(\underbrace{\mathbf{A} \cdot \mathbf{s}}_{=\mathbf{b}}) = \mathbf{m} \ . \end{aligned}$$

Malheureusement, Bad-LWE est terriblement vulnérable, car « l'algèbre linéaire est facile ». Par exemple, effectuons une attaque par texte chiffré choisi en chiffrant toujours 0. Nous observons des **échantillons :**

$$\mathbf{rA}, (\mathbf{rA}) \cdot \mathbf{s} \tag{31}$$

pour des $\mathbf{r}$ et $\mathbf{s}$ inconnus. Après avoir collecté suffisamment d'échantillons, nous les plaçons en lignes dans une matrice $\mathbf{R}$ que nous espérons être inversible (une matrice aléatoire a de

grandes chances d'être inversible) : nous avons $\mathbf{R}$ et $\mathbf{Rs}$, il suffit donc d'inverser $\mathbf{R}$ pour récupérer le secret.

5.2 Le Problème LWE

LWE peut être vu comme une manière de rendre le cryptosystème ci-dessus sûr, en introduisant des erreurs qui transforment le problème d'algèbre linéaire (facile) en un problème fondamentalement difficile.

Commençons par définir ce qu'est une distribution Gaussienne.

Définition 14. Let $s > 0, c \in \mathbb{R}^n, x \in \mathbb{R}^n$, we define $\rho_{s,c}(x) = e^{-\pi\|x-c\|^2/s^2}$ and $D_{s,c}(x) = \rho_{s,c}(x)/s^n$. $D_{s,c}$ is the density of probability of the Gaussian distribution of center c and variance $ns^2/(2\pi)$ (of parameter s).

Lemme 6. *Let $s > 0$,* $\Pr_{x \leftarrow D_s}[\|x\| \geq \sqrt{n}s] \leq 2^{-n}$.

Cela signifie qu'un élément échantillonné selon une distribution Gaussienne est petit (de norme plus petite que ns) avec très forte probabilité. On peut aussi additionner facilement des Gaussiennes discrètes.

Définitions. La distribution LWE consiste à prendre des produits scalaires entre le secret $\mathbf{s} \in \mathbb{Z}_q^n$ et des vecteurs aléatoires uniformes $\mathbf{a}_i \in \mathbb{Z}_q^n$, puis à **bruiter** ces résultats à l'aide d'une **petite** erreur, qui est tirée à l'aide d'une distribution Gaussienne avec un écart-type « faible » .

Définition 15. Let $n \geq 1, q \geq 2$ prime, $0 < \alpha < 1$. The distribution $D_{n,q,\alpha}^{LWE}(\mathbf{s})$ is the discrete distribution over $\mathbb{Z}_q^{n+1}$ obtained by :

1. Sample $\mathbf{a} \hookleftarrow U(\mathbb{Z}_q^n)$
2. $e \hookleftarrow D_{\mathbb{Z}^\ell,\alpha}$
3. Return $(\mathbf{a}, (\mathbf{a} \cdot \mathbf{s}) + e \mod q)$

Le problème search-LWE consiste à retrouver le secret $\mathbf{s}$. Le problème LWE décisionnel consiste à distinguer la distribution LWE de l'uniforme.

Problème 5 (Search $LWE_{n,q,\alpha}$)**.** *Let $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$. Given samples from $D_{n,q,\alpha}^{LWE}(\mathbf{s})$, find $\mathbf{s}$.*

Problème 6 (Decisional LWE)**.** *Let $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$. Distinguish between $D_{n,q,\alpha}^{LWE}(\mathbf{s})$ and $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$.*

Most of the time we choose q polynomial in n, and α between $1/q$ and 1.

If α is very small, then with probability very close to 1, the error e of a sample will be 0. In that case, from n samples, we obtain a linear system in $\mathbf{s}$, that we can solve in polynomial time.

If α is big, e is very close from uniform modulo q. The distribution $D_{n,q,\alpha}^{LWE}(\mathbf{s})$ is close to uniform (independently from $\mathbf{s}$), and finding $\mathbf{s}$ is impossible.

Réduction.

Lemme 7. *Lorsque q est polynomial en n, search-LWE et decision-LWE sont équivalents en termes de complexité computationnelle.*

▷ **preuve en TD**

Démonstration. **Réduction de Décision à Recherche.** Supposons que nous avons un algorithme $\mathcal{A}$ qui résout le problème de recherche, nous l'utilisons pour résoudre le problème de décision.

En entrée du problème de décision, nous recevons un ensemble d'échantillons : $\mathbf{a}_i, b_i$ où soit $b_i = \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i$, soit b_i est uniforme et indépendant de $\mathbf{a}_i$.

Notre stratégie est la suivante : nous prenons ces échantillons et les donnons à la boîte noire de recherche. La boîte noire de recherche renvoie une valeur $\mathbf{s}$. Nous vérifions si $\mathbf{b} - \mathbf{As}$ est court. Si c'est le cas, nous disons que les échantillons sont de type LWE. Sinon, nous disons qu'ils sont aléatoires.

Cette procédure réussit-elle, et avec quelle probabilité ? L'idée est la suivante. Regardons d'abord le cas LWE. Dans ce cas, l'algorithme de recherche réussira. À son tour, notre recomputation de $\mathbf{b}-\mathbf{As}$ renverra en effet un vecteur court. Par conséquent, nous réussissons à reconnaître le cas LWE.

Ensuite, regardons le cas aléatoire. Dans ce cas, l'algorithme de recherche échouera. Il renverra un certain vecteur $\mathbf{s}$. En calculant $\mathbf{b} - \mathbf{As}$, nous obtenons un vecteur aléatoire dans $\mathbb{Z}_q^n$. Il est très peu probable que ce vecteur soit court. Donc notre procédure réussit avec une probabilité écrasante.

Réduction de la Recherche à la Décision. Nous essayons de récupérer la première coordonnée de $\mathbf{s}$, en testant si elle est égale à une valeur k choisie. Comme q est polynomial en n, nous pouvons le faire pour tous les k, puis pour toutes les coordonnées, et ainsi nous aurons trouvé $\mathbf{s}$.

Une remarque importante ici est que nous avons besoin que la routine de décision soit fiable, car nous l'appellerons plusieurs fois. Pour rendre sa probabilité d'erreur très faible, une méthode typique consiste à la faire fonctionner plusieurs fois et à prendre la majorité des résultats. Cela sera suffisant pour nous, et nous pouvons supposer à partir de maintenant que la routine de décision réussit à chaque appel.

Afin d'appeler la routine de décision, nous modifions nos entrées de manière à ce que : si nous avons deviné k correctement, elles seront de type LWE, sinon, elles seront uniformément aléatoires. Pour cela, nous devons introduire notre propre aléa dans ces échantillons.

Pour chaque échantillon LWE $\mathbf{a_i}, b_i = \mathbf{a_i} \cdot \mathbf{s} + e_i$, nous le modifions comme suit : nous choisissons une nouvelle valeur r uniformément aléatoire, nous ajoutons $(r, 0, \ldots, 0)$ à $\mathbf{a_i}$ et nous ajoutons $(rk) \mod q$ à b_i. Ensuite :

- si $s_0 = k$, l'échantillon modifié est de la forme $\mathbf{a_i}', \mathbf{a_i}' \cdot \mathbf{s} + e_i$ où $\mathbf{a_i}' = \mathbf{a_i} + (r, 0, \ldots, 0)$ est *encore uniformément aléatoire* (ce qui est important pour notre procédure de décision !)
- si $s_0 \neq k$, l'échantillon modifié est $\mathbf{a_i}', b_i'$ où $b_i' = \mathbf{a_i} \cdot \mathbf{s} + e_i + (rk) = \mathbf{a_i}' \cdot \mathbf{s} + r(k - s_0) + e_i$. Ici, k et s_0 sont des constantes, donc $r(k - s_0)$ est uniformément aléatoire dans $\mathbb{Z}_q$ et indépendant de $\mathbf{a}_i'$. Cela signifie que dans ces nouveaux échantillons, $\mathbf{a}_i'$ est uniformément aléatoire, et b_i' l'est également (il devient non corrélé à $\mathbf{a_i}'$, pour ainsi dire.)

Ainsi, notre procédure de décision peut distinguer dans quel cas nous nous trouvons.

Une dernière petite étape : en raison de la définition du problème de LWE décisionnel, nous avons supposé que $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$ avant d'exécuter cet algorithme. Nous pouvons montrer que si nous pouvons récupérer $\mathbf{s}$ lorsqu'il est aléatoire, alors nous pouvons également trouver **n'importe quel $\mathbf{s}$**.

En effet, lorsque $\mathbf{s}$ est fixé, nous pouvons échantillonner $\mathbf{t} \hookleftarrow U(\mathbb{Z}_q^n)$ et transformer : $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + e)$ en $(\mathbf{a}, \mathbf{a} \cdot (\mathbf{s} + \mathbf{t}) + e)$ pour le secret $\mathbf{s} + \mathbf{t}$, qui devient uniforme. Nous pouvons récupérer $\mathbf{s} + \mathbf{t}$ et donc $\mathbf{s}$. □

Difficulté de LWE. Du point de vue algorithmique, LWE semble extrêmement difficile à résoudre. Le meilleur algorithme connu pour des paramètres (n, q, α) généraux repose sur la réduction des réseaux euclidiens. Ce lien avec les réseaux euclidiens n'est pas anodin. En effet, les algorithmes sur les réseaux offrent les meilleures attaques contre LWE, mais, par ailleurs, on peut montrer que s'il existait un algorithme efficace résolvant LWE, alors

il existerait des algorithmes efficaces contre des problèmes réputés difficiles portant sur les réseaux.

Plus précisément, l'existence d'un algorithme efficace résolvant LWE (pour une erreur gaussienne) implique l'existence d'un algorithme efficace pour résoudre GapSVP_γ.

Théorème 5. *Let $q \geq 2$ and $\alpha \in (0,1)$ functions of n such that $\alpha q \geq 2\sqrt{n}$. If q is prime and polynomial in n, there exists a quantum polynomial reduction from $GapSVP_\gamma$ in dimension n to $LWE_{n,q,\alpha}$ with $\gamma = \widetilde{\mathcal{O}}(n/\alpha)$. For all q there exists a classical polynomial reduction from $GapSVP_\gamma$ in dimension $\Theta(\sqrt{n})$ to $LWE_{n,q,\alpha}$ with $\gamma = \widetilde{\mathcal{O}}\big(n^2/\alpha\big)$.*

5.3 Chiffrement de Regev

Le premier chiffrement dont la sécurité repose sur la difficulté du problème LWE a été proposé par Regev en 2005 [Reg05]. La dimension n est son paramètre de sécurité.

Soit n, ℓ, q des entiers avec q premier et $\ell \geq 4(n+1)\log_2 q$ et $\alpha \in]0, 1/(8\ell)[$.

Parce que LWE introduit des erreurs, nous avons besoin d'une manière de les supprimer lors du déchiffrement. Nous formalisons cela avec une paire de fonctions Compress et Decompress que nous utiliserons dans ce qui suit. Compress prend un entier modulo q et le décode en 0 s'il est plus proche de 0, ou en 1 s'il est plus proche de $\lfloor q/2 \rfloor$. Decompress fait l'inverse : il encode 0 en 0 et 1 en $\lfloor q/2 \rfloor$.

$$\begin{cases} \mathsf{Compress} : & \mathbb{Z}_q & \to & \{0,1\} \\ & u & \mapsto & \mathsf{round}\left(u/\lfloor q/2 \rfloor\right) \pmod 2 \end{cases}$$

$$\begin{cases} \mathsf{Decompress} : & \{0,1\} & \to & \mathbb{Z}_q \\ & m & \mapsto & m\lfloor q/2 \rfloor \end{cases}$$

Remarque 7. Compress et Decompress fonctionnent **modulo** q, et de cette perspective, tout ce qui est proche de q (par exemple, $q-1, q-2$) est "petit". Par exemple : $\mathsf{Compress}(q-1) = 0$. Plus généralement, vous devez penser à ces nombres modulo q comme étant représentés par un entier compris entre $-\lfloor q/2 \rfloor$ et $\lfloor q/2 \rfloor$.

Nous pouvons maintenant introduire le schéma de chiffrement à clé publique de Regev. Il est sécurisé, mais peu efficace, car il ne chiffre qu'un seul bit et nécessite une clé de taille $\mathcal{O}(n^2)$.

LWE PKE

KeyGen :

- Clé privée : $\mathbf{s} \in \mathbb{Z}_q^n$ aléatoire
- Clé publique : $(\mathbf{A}, \mathbf{b} := \mathbf{A}\mathbf{s} + \mathbf{e})$ où $\mathbf{A}$ est une matrice aléatoire $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$, et $\mathbf{e} \in \mathbb{Z}_q^\ell$ est échantillonné en utilisant la distribution d'erreurs "petites" (i.e., Gaussienne discrète)

Enc $\mathbf{m} \in \{0,1\}$:

- Choisir un vecteur aléatoire $\mathbf{r} \in \{0,1\}^\ell$
- Retourner $\mathbf{c_1}, \mathbf{c_2} := \mathbf{r}\mathbf{A}, (\mathsf{Decompress}(\mathbf{m}) + \mathbf{r} \cdot \mathbf{b})$

Dec $\mathbf{c} = (\mathbf{c_1}, \mathbf{c_2}) \in \mathbb{Z}_q^{n+1}$:

- $\mathbf{m} = \mathsf{Compress}(\mathbf{c_2} - \mathbf{c_1} \cdot \mathbf{s})$

Pour analyser le déchiffrement, calculons :

$$\begin{aligned}\mathbf{c_2} - \mathbf{c_1} \cdot \mathbf{s} &= (\mathsf{Decompress}(\mathbf{m}) + \mathbf{r} \cdot \mathbf{b}) - (\mathbf{rA}) \cdot \mathbf{s} \\ &= \mathsf{Decompress}(\mathbf{m}) + \mathbf{r}(\mathbf{As} + \mathbf{e}) - \mathbf{rAs} \\ &= \mathsf{Decompress}(\mathbf{m}) + \underbrace{\mathbf{r} \cdot \mathbf{e}}_{\text{Petit}}\end{aligned}$$

On a choisi $\mathbf{r}$ uniforme dans $\{0,1\}^\ell$, et $\mathbf{e}$ suit une Gaussienne discrète de paramètre $\alpha q \leq q/(8m)$, donc avec probabilité proche de 1 :

$$|\sum r_i e_i| \leq \|\mathbf{r}\| \cdot \|\mathbf{e}\| \leq \sqrt{m}\frac{q}{8\sqrt{m}} = \frac{q}{8} \ . \tag{32}$$

En appelant $\mathsf{Compress}$ sur ce résultat on devrait donc bien retrouver la valeur de m. Cependant, en fonction de notre choix de distribution d'erreur, il peut y avoir une certaine probabilité que l'erreur soit trop grande et que le déchiffrement échoue. Cette probabilité d'échec apparaît même dans les schémas modernes tels que Kyber. Il suffit de s'assurer qu'elle est assez faible pour ne jamais se produire dans la pratique.

Sécurité. La preuve de sécurité du chiffrement de Regev repose sur le lemme suivant, qui est un cas particulier du *leftover hash lemma*. De manière plus générale, le lemme montre que si un adversaire obtient une information partielle sur une valeur de n bits (par exemple t bits d'information), il est toujours possible d'extraire $n - t$ bits sur lesquels il n'aura aucune information.

Lemme 8 (Leftover Hash Lemma)**.** *Soit $\ell, n, q \geq 1$ des entiers tels que $\ell \geq 4n \log q$, q est premier, et $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{\ell \times n})$ et $\mathbf{r} \hookleftarrow U(\{0,1\}^\ell)$. Alors $(\mathbf{A}, \mathbf{rA})$ est à distance statistique $\leq 2^{-n}$ de la distribution uniforme sur $\mathbb{Z}_q^{\ell \times n} \times \mathbb{Z}_q^n$.*

Lemme 9. *Si LWE est difficile, le schéma de Regev est IND-CPA.*

Démonstration. Nous montrons que tout adversaire PPT $\mathcal{A}$ qui attaque la propriété IND-CPA du schéma de Regev a un avantage négligeable. Cela repose sur trois jeux G_0, G_1, G_2 où G_0 est le jeu IND-CPA.

Nous allons montrer qu'un adversaire PPT ne peut distinguer entre G_0 et G_1, puis entre G_1 et G_2 (à probabilité négligeable près), puis que son avantage dans le jeu G_2 est nul. Ce qui nous donnera le résultat.

Algorithm 3 Jeu G_0(IND-CPA)

1: $\mathsf{pk} \leftarrow \mathsf{KeyGen}(n)$ i.e. $\mathsf{pk} = (\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e})$ choisi selon la distribution LWE
2: $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$
3: $b' \leftarrow U(\{0,1\})$
4: $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{b'})$ i.e. $C^* = (\mathbf{rA}, \mathbf{rb} + \lfloor q/2 \rfloor m_{b'})$ où $\mathbf{r}$ est uniforme
5: $b'' \leftarrow \mathcal{A}(C^*)$
6: Renvoyer 1 si $b' = b''$, 0 sinon

L'avantage de $\mathcal{A}$ dans le jeu G_0 est :

$$\mathrm{Adv}^{IND-CPA}(\mathcal{A}) = |\Pr[b'' = b'] - 1/2| = |\Pr_{G_0}[\mathcal{A} \text{ Wins}] - 1/2| \ . \tag{33}$$

Jeu G1. Nous modifions la génération de la clé publique dans le jeu G_0. Au lieu de la vraie distribution LWE, la deuxième partie devient uniforme : $\mathbf{b} \hookleftarrow U(\mathbb{Z}_q^n)$. La clé publique

Algorithm 4 Jeu G1

1: $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$ où $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{\ell \times n})$ et $\mathbf{b} \hookleftarrow U(\mathbb{Z}_q^n)$
2: $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$
3: $b' \hookleftarrow U(\{0,1\})$
4: $C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{b'})$ i.e. $C^* = (\mathbf{rA}, \mathbf{rb} + \lfloor q/2 \rfloor m_{b'})$ où $\mathbf{r} \hookleftarrow U(\{0,1\}^\ell)$
5: $b'' \leftarrow \mathcal{A}(C^*)$
6: Renvoyer 1 si $b' = b''$, 0 sinon

Sous l'hypothèse LWE (décisionnel), les jeux G_0 et G_1 sont indistinguables. En d'autres termes, la différence des probabilités qu'a l'attaquant $\mathcal{A}$ de renvoyer 1 dans les deux jeux est négligeable.

$$\left| \Pr_{G_0}[\mathcal{A} \text{ Wins}] - \Pr_{G_1}[\mathcal{A} \text{ Wins}] \right| = \text{negl}(n) \quad . \tag{34}$$

(Autrement, nous pourrions utiliser $\mathcal{A}$ comme distingueur pour LWE).

Jeu 2. Nous modifions maintenant la génération du chiffré challenge dans G_1. Nous remplaçons :

$$C^* = (\mathbf{rA}, \mathbf{rb} + \lfloor q/2 \rfloor m_{b'})$$

par une valeur aléatoire : $C^* \leftarrow U(\mathbb{Z}_q^{n+1})$.

Algorithm 5 Jeu G2

1: $\mathsf{pk} = (\mathbf{A}, \mathbf{b})$ où $\mathbf{A} \hookleftarrow U(\mathbb{Z}_q^{\ell \times n})$ et $\mathbf{b} \hookleftarrow U(\mathbb{Z}_q^n)$
2: $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$
3: $b' \hookleftarrow U(\{0,1\})$
4: $C^* \leftarrow U(\mathbb{Z}_q^{n+1})$
5: $b'' \leftarrow \mathcal{A}(C^*)$
6: Renvoyer 1 si $b' = b''$, 0 sinon

Regardons ce que l'adversaire reçoit dans les deux cas. Dans G_1, on a déjà remplacé $\mathbf{b}$ par une valeur aléatoire uniforme indépendante de $\mathbf{A}$, par conséquent le chiffré C^* est $(\mathbf{rA}, u)$ où u et $\mathbf{rA}$ sont uniformes et indépendants. De plus l'adversaire connaît $\mathbf{A}$.

Dans G_2, on a remplacé C^* par une valeur uniforme. Cela ne change rien pour la partie droite. Pour la partie gauche, la différence de probabilité est bornée par la distance statistique entre $(\mathbf{A}, \mathbf{rA})$ et $(\mathbf{A}, \mathbf{U})$, elle-même négligeable d'après le LHL. Par conséquent :

$$\left| \Pr_{G_1}[\mathcal{A} \text{ Wins}] - \Pr_{G_2}[\mathcal{A} \text{ Wins}] \right| = \text{negl}(n) \quad . \tag{35}$$

Et pour finir, on note que dans G_2, $\mathcal{A}$ reçoit un chiffré challenge complètement aléatoire. Par conséquent on a : $\Pr_{G_2}[\mathcal{A} \text{ Wins}] = \frac{1}{2}$. On conclut par :

$$\begin{aligned}
&\text{Adv}^{IND-CPA}(\mathcal{A}) = |\Pr_{G_0}[\mathcal{A} \text{ Wins}] - 1/2| \\
&\leq \underbrace{|\Pr_{G_0}[\mathcal{A} \text{ Wins}] - \Pr_{G_1}[\mathcal{A} \text{ Wins}]|}_{\text{negl}(n) \text{ (LWE)}} + \underbrace{|\Pr_{G_1}[\mathcal{A} \text{ Wins}] - \Pr_{G_2}[\mathcal{A} \text{ Wins}]|}_{\text{negl}(n) \text{ (statistical)}} + \underbrace{|\Pr_{G_2}[\mathcal{A} \text{ Wins}] - 1/2|}_{=0} \\
&\leq \text{negl}(n) \quad .
\end{aligned}$$

□

Résumé

- LWE est l'hypothèse de sécurité fondamentale de la cryptographie basée sur les réseaux euclidiens, la branche la plus importante de la **cryptographie post-quantique**
- Le chiffrement basé sur Regev (ne pas utiliser en pratique !) est IND-CPA sous l'hypothèse LWE. Il n'est pas IND-CCA.

Références

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In : *STOC*. ACM, 2005, p. 84-93.

6 Signatures Numériques

Jusqu'ici nous avons vu des constructions de **chiffrement**, qui n'apportent que de la **confidentialité**. Ce n'est absolument pas suffisant en pratique. Grâce à une **signature numérique**, on peut apporter deux autres garanties :

- Authenticité : je suis bien l'auteur d'un message ;
- Intégrité : le message n'a pas été modifié.

La clé publique est maintenant la clé de **vérification**. Pour signer un document, j'utilise ma **clé secrète**, puis j'envoie le message et sa signature. N'importe qui peut vérifier que je suis bien l'auteur du message. Pour être exact, la signature prouve que je possède la clé secrète associée.

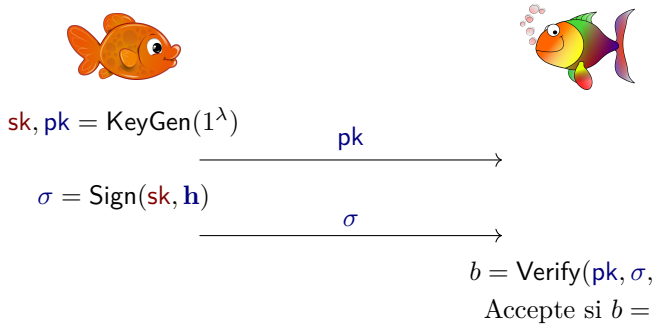
Le scénario d'attaque le plus courant est celui dans lequel l'adversaire compromet cette authenticité, en construisant une signature contrefaite (*forgery*) sans connaître la clé secrète.

6.1 Définitions

Définition 16 (Signature numérique (*digital signature*))**.** Une signature numérique est un triplet d'algorithmes polynomiaux :

$$\begin{cases} \mathsf{KeyGen} : & 1^n & \mapsto & \mathsf{sk}, \mathsf{pk} \\ \mathsf{Sign} : & \mathsf{sk}, \mathbf{h} & \mapsto & \sigma \\ \mathsf{Verify} : & \mathsf{pk}, \sigma, \mathbf{h} & \mapsto & \{0,1\} \end{cases}$$

tels que pour tout m, $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(m, \mathsf{sk})) = 1$.



Notez bien que cette définition ne protège pas la confidentialité du « message ». En réalité, on n'utilise jamais un schéma de signature pour signer un document ; seulement le **haché** en utilisant une fonction de hachage cryptographique. La plupart du temps cette étape de hachage est incluse dans le standard, et on appelle cela un schéma de signature *hash-and-sign.*

La principale notion de sécurité pour un schéma de signature est la sécurité EUF-CMA (*existential unforgeability against chosen-message attacks*). Elle est définie par le jeu de sécurité suivant entre $\mathcal{C}$ et $\mathcal{A}$.

1. Initialisation : le challenger génère $\mathsf{pk}, \mathsf{sk} \leftarrow \mathsf{KeyGen}(n)$ et envoie pk à $\mathcal{A}$
2. Requêtes : à tout moment, $\mathcal{A}$ peut envoyer un message m_i à $\mathcal{C}$, et récupérer $\sigma_i = \mathsf{Sign}(\mathsf{sk}, m_i)$
3. Contrefaçon (forge) : $\mathcal{A}$ renvoie une paire m^*, σ^*. Il gagne si $\mathsf{Verify}(\mathsf{sk}, m^*, \sigma^*) = 1$ et $m^* \neq m_i$.

En d'autres termes, l'adversaire doit parvenir à signer un message dont il n'a pas précédemment demandé la signature. Il existe une variante *strong EUF-CMA* dans laquelle

le message peut être le même qu'un des messages précédents, mais la signature doit être une nouvelle $(\forall i, (m^*, \sigma^*) \neq (m_i, \sigma_i))$.

L'avantage est défini par :

$$\mathrm{Adv}^{EUF-CMA}(\mathcal{A}) = |\Pr[\mathcal{A} \text{ win}]| \ . \tag{36}$$

Un schéma de signature est dit sûr pour la sécurité EU-CMA si tout adversaire PPT a un avantage négligeable dans ce jeu de sécurité.

6.2 Textbook RSA Signature

Principe : on "échange" l'utilisation des clés par rapport à un chiffrement à clé publique.

RSA Signature

KeyGen :

- Choisir les premiers P, Q, $N = PQ$, e, d tels que $ed = 1 \pmod{\phi(N)}$.
- $\mathsf{sk} = (N, d)$
- $\mathsf{pk} = (N, e)$

Sign $m \in \mathbb{Z}_N^*$

- Renvoyer $\sigma = m^d \pmod{N}$

Verify $m \in \mathbb{Z}_N^*, \sigma$

- Vérifier que $\sigma^e = m \pmod{N}$

Ce n'est pas sûr. Prenons σ quelconque et calculons $m = \sigma^e \pmod{N}$. La paire (m, σ) est une paire (message,signature) valide. Certes, on ne contrôle pas m, mais ce n'est pas exigé dans le jeu EUF-CMA.

On peut aller plus loin et contrefaire une signature d'un message donné $m \in \mathbb{Z}_N^*$. L'adversaire envoie deux requêtes de signature sur $m_1 \in \mathbb{Z}_N^*$ et $m_2 = m(m_1)^{-1} \mod N$ et reçoit σ_1, σ_2. Alors $\sigma = \sigma_1\sigma_2$ est une signature valide de m.

6.3 Hash-and-Sign RSA

Dans cette section nous décrivons "Hash-and-sign RSA" qui est prouvé dans le modèle de l'oracle aléatoire (ROM), c'est-à-dire en modélisant la fonction de hachage par une fonction aléatoire idéale.

Hash-and-sign RSA Signature

On suppose disposer d'une fonction de hachage cryptographique $H : \{0,1\}^* \to \mathbb{Z}_N^*$.
KeyGen :

- Choisir P, Q premiers, $N = PQ$, e, d tels que $ed = 1 \pmod{\phi(N)}$.
- $\mathsf{sk} = (N, d)$
- $\mathsf{pk} = (N, e)$

Sign $m \in \{0,1\}^*$

- Renvoyer $\sigma = H(m)^d \mod N$

Verify $m \in \{0,1\}^*, \sigma$

- Vérifier que $\sigma^e = H(m) \mod N$

Nous n'allons pas faire la preuve, mais notez que les deux attaques précédentes ne sont plus possibles.

- En prenant une signature quelconque, et en calculant σ^e, il nous faudrait maintenant trouver m tel que $H(m) = \sigma^e$: c'est une recherche de préimage (précisément un problème difficile pour une fonction de hachage cryptographique).
- Comme on passe par H, les signatures ne sont plus malléables.

Résumé

- Une signature numérique apporte **authenticité** et **intégrité**
- La sécurité d'une signature est basée sur un jeu de contrefaçon (*unforgeability*)

7 Cryptographie Symétrique

La cryptographie symétrique est le contexte dans lequel Alice et Bob disposent de la même information. Ce qui inclut :

- Le cas où ils ont tous les deux une clé partagée (chiffrement symétrique) ;
- Le cas où il n'y a pas de clé du tout (fonctions de hachage).

Bien que ces deux situations soient tout à fait distinctes, les constructions modernes de fonctions de hachage et de chiffrement symétrique utilisent des briques de bases très proches.

Les **hypothèses de sécurité** utilisées en cryptographie symétrique sont très différentes de la cryptographie asymétrique. Elles sont plus *ad hoc* et reposent moins sur des problèmes mathématiques généraux. Pour autant, la notion de sécurité demeure conservatrice, et la cryptanalyse occupe une part très importante du travail des cryptographes.

7.1 Chiffrement à Bloc

Définition 17. Soit X un ensemble et $\mathcal{K}$ un ensemble de clés. Un **chiffrement à bloc** E est une famille de fonctions de X dans X, indexée par $\mathcal{K}$, telle que :

- Pour toute clé $K \in \mathcal{K}$, la fonction E_K est une permutation de X
- Pour toute clé $K \in \mathcal{K}$, E_K et son inverse E_K^{-1} sont calculables en temps polynomial.

En général on prend $\mathcal{K} = \{0,1\}^k$, où k est la taille de clé, et $X = \{0,1\}^n$, où n est la taille de bloc.

Si dans les sections précédentes, n et k devaient être vus comme des paramètres variables, dans le monde de la cryptographie symétrique il s'agira le plus souvent de constantes. Un chiffrement à bloc n'a pas vocation à prendre une taille gigantesque. Des valeurs de $n = 128$ et $k = 128$ ou 256 sont typiques.

Sécurité. La principale notion de sécurité pour un chiffrement à bloc est celle de permutation pseudo-aléatoire (PRP, *pseudo-random permutation*).

Un adversaire $\mathcal{D}$ (distingueur PRP) joue au jeu suivant avec un challenger $\mathcal{C}$:

- $\mathcal{C}$ tire une clé au hasard $K \hookleftarrow U(\mathcal{K})$ et un bit $b \hookleftarrow U(0,1)$, et une permutation aléatoire Π sur $\{0,1\}^n$.
  Dans la suite, $\mathcal{C}$ répondra aux requêtes de chiffrement et de déchiffrement de $\mathcal{D}$ de la manière suivante :
  - Cas $b = 0$ (RAND) : $\mathcal{C}$ répond en utilisant Π ou Π^{-1}
  - Cas $b = 1$ (PRP) : $\mathcal{C}$ répond en utilisant E_K ou E_K^{-1}
- $\mathcal{D}$ calcule un bit b', le renvoie, et gagne si $b' = b$.

L'avantage est défini par :

$$\mathrm{Adv}^{PRP}(\mathcal{D}) = |\Pr[\mathcal{D} \text{ Win}] - 1/2| \ . \tag{37}$$

Dans ce jeu, $\mathcal{D}$ doit donc distinguer un chiffrement avec une clé aléatoire d'une permutation aléatoire quelconque. La PRP est dite *strong PRP* lorsqu'il a accès à la fois à la permutation et son inverse.

Sécurité Générique. Pour tout chiffrement à bloc, on peut réussir le jeu en essayant toutes les clés possibles : cela prend un temps 2^K.

De manière plus générale, on étudie les chiffrements à bloc sous l'angle des attaques par **recouvrement de clé**, à **clair choisi** et à **chiffré choisi**. Dans ce cas en faisant quelques requêtes de chiffrement, on dispose d'assez d'information pour déterminer si une clé donnée est la bonne ou non, et il suffit donc de toutes les tester.

Exemple 4. Le chiffrement à bloc AES offre trois tailles de clés 128, 192 et 256 bits. L'attaque par force brute demande donc de l'ordre de 2^{128}, 2^{192}, 2^{256} calculs (évaluations d'AES) en moyenne.

En cryptographie symétrique, un chiffrement à bloc est considéré comme **sûr** s'il n'existe pas d'algorithme (connu) permettant de retrouver la clé (ou plus généralement, de résoudre le jeu PRP) en moins de 2^K opérations. Si au contraire un tel algorithme existe, il est appelé une **attaque** ; le chiffrement est **cassé** et n'est plus utilisé. Cela signifie que si l'on pouvait retrouver la clé de l'AES-128 en temps 2^{120}, le chiffrement serait considéré comme cassé et non sûr, même si 2^{120} demeure infaisable [1].

L'une des raisons est l'absence de paramétrisation : les tailles de clé et de bloc sont des paramètres fixes du standard. De plus, la découverte d'une attaque conduit généralement à des améliorations successives qui peuvent rendre la procédure faisable en pratique. Les exigences de sécurité des primitives symétriques sont donc volontairement **conservatrices**.

7.2 Fonction Pseudo-Aléatoire

Une fonction pseudo-aléatoire (PRF) a exactement la définition d'un chiffrement à bloc, dans laquelle on aurait enlevé l'inversibilité.

Définition 18. Une PRF F est une famille de fonctions de $\{0,1\}^n$ dans lui-même, indexée par $\mathcal{K} = \{0,1\}^k$, telle que l'avantage d'un distingueur PPT dans le jeu de sécurité PRF est négligeable.

Le jeu de sécurité PRF consiste simplement à distinguer F_K (où K est tirée au hasard) d'une fonction aléatoire.

Il s'avère que dans le contexte des modes d'opération (voir plus bas), on utilise souvent un chiffrement à bloc comme une PRF. Dans ce contexte, beaucoup de preuves de sécurité reposent sur le lemme suivant.

Lemme 10 (PRP-PRF switching)**.** *Soit $\mathcal{A}$ un adversaire faisant q requêtes et F_K une famille de fonctions pseudo-aléatoires sur $\{0,1\}^n$, alors :*

$$\left|\mathrm{Adv}^{PRF}(\mathcal{A}) - \mathrm{Adv}^{PRP}(\mathcal{A})\right| \leq \frac{q^2}{2^{n+1}} \ . \tag{38}$$

(Pour le jeu weak-PRP, sans appels à la permutation inverse).

Démonstration. Supposons que nous exécutons $\mathcal{A}$ dans le jeu PRF, et soit E l'évènement selon lequel le challenger renvoie la même valeur pour deux entrées distinctes. Nous avons :

$$\begin{aligned}
\Pr[\mathcal{A} \text{ gagne PRF}] &= \Pr[\mathcal{A} \text{ gagne PRF}|E]\Pr[E] \\
&\quad + \Pr[\mathcal{A} \text{ gagne PRF}|\neg E]\Pr[\neg E] \\
&\leq \Pr[E] + \Pr[\mathcal{A} \text{ gagne PRF}|\neg E] \\
&= \Pr[E] + \Pr[\mathcal{A} \text{ gagne PRP}|\neg E] \\
&\leq \Pr[E] + \Pr[\mathcal{A} \text{ gagne PRP}] \ .
\end{aligned}$$

En effet, si l'évènement E ne se produit pas, la vue de l'adversaire est équivalente au jeu PRP (une fonction sans collision est une permutation !).

Maintenant, on borne la probabilité que E aie lieu : c'est $q^2/(2^{n+1})$ par la borne des anniversaires. Par conséquent, en supposant que la probabilité de $\mathcal{A}$ de gagner est toujours

1. Il y a parfois quelques nuances à ces définitions. L'important est surtout que les conjectures de sécurité soient claires.

au moins 1/2 (sans perte de généralité) :

$$\begin{aligned}\left|\mathrm{Adv}^{PRF}(\mathcal{A}) - \mathrm{Adv}^{PRP}(\mathcal{A})\right| &= |\ |\Pr[\mathcal{A} \text{ gagne PRF}] - 1/2| - |\Pr[\mathcal{A} \text{ gagne PRP}] - 1/2|\ | \\ &= |\Pr[\mathcal{A} \text{ gagne PRF}] - \Pr[\mathcal{A} \text{ gagne PRP}]| \\ &\leq \Pr[E] = \frac{q^2}{2^{n+1}}\ .\end{aligned}$$

□

Cette borne de sécurité se retrouve dans la sécurité de nombreux modes d'opération (CBC, CTR...) qui sont sûrs tant que le nombre de blocs chiffrés demeure très inférieur à $2^{n/2}$. C'est-à-dire, 2^{32} quand on utilise un chiffrement à bloc de 64 bits, ou 2^{64} pour un bloc de 128 bits (comme AES).

7.3 Modes d'Opération

Un chiffrement à bloc seul ne peut pas être utilisé pour chiffrer. Toutefois, on peut s'en servir pour définir un **mode d'opération** (*operation mode*) qui permettra de chiffrer un message de taille quelconque. Formellement, ce mode d'opération définit un **chiffrement symétrique** prenant en entrée des messages de taille quelconque. (En réalité, il y a toujours une limitation sur la taille du message, après laquelle il est nécessaire d'utiliser une nouvelle clé ; mais celle-ci est très large).

Nous supposons dans un premier temps que les messages sont donnés par blocs. Pour des messages de taille quelconque, il faut utiliser un **remplissage** (détaillé plus bas avec Merkle-Damgård).

ECB. Il est d'usage dans un cours d'introduction à la cryptographie de donner le mode ECB (*electronic codebook*) et d'expliquer doctement qu'il n'est pas très sûr. Nous ne ferons même pas l'effort de dire pourquoi.

CBC Le mode CBC (cipher block chaining) (figure 2) consiste à chaîner les blocs.

CBC avec IV aléatoire

$\mathsf{Enc}(K, m)$:

- Séparer m en blocs $m_0, \ldots, m_{t-1}$ de taille n (avec remplissage si nécessaire)
- $IV \hookleftarrow U(\{0,1\}^n)$
- $c_0 \leftarrow E_K(m_0 \oplus IV)$
- For $i = 1$ to $t-1$: $c_i \leftarrow E_K(c_{i-1} \oplus m_i)$
- Return $IV, c_0, \ldots, c_{t-1}$

$\mathsf{Dec}(K, IV, c)$:

- $m_0 \leftarrow E_K^{-1}(c_0) \oplus IV$
- For $i = 1$ to $t-1$: $m_i \leftarrow E_K^{-1}(c_i) \oplus c_{i-1}$

Théorème 6. *CBC avec une IV aléatoire est IND-CPA si le chiffrement à bloc est une PRP :*

$$\mathrm{Adv}^{CPA}(\mathcal{A}) \leq \mathrm{Adv}^{PRP}(\mathcal{B}) + \sigma^2/2^{n-1} \tag{39}$$

où σ est le nombre total de blocs chiffrés dans les requêtes à CBC.

On ne fera pas la preuve complète car elle est un peu technique, mais on notera qu'elle fait intervenir le PRP-PRF switching, d'où le terme quadratique. Il est intéressant de remarquer que si l'on remplace l'IV par une « nonce » (un nombre unique, qui change à chaque requête), CBC n'est plus IND-CPA.

CTR (Counter Mode). Le mode CTR consiste à partir d'une IV, à incrémenter sa valeur, et à utiliser les blocs successifs comme une **suite chiffrante** que l'on XOR aux blocs de message. Cela vous rappellera sans doute le One-time pad.

Comme CBC, CTR peut s'utiliser avec une IV aléatoire. Nous présentons ici une version avec un « compteur » où l'IV est remplacée par une variable interne *ctr* (compteur) qui commence à 0, est incrémentée à chaque bloc, et conserve son état. Cette version rend la preuve de sécurité plus facile, car avec un compteur, il ne peut pas y avoir de collisions intempestives entre les IV (autrement il faudrait tenir compte de cela dans la preuve).

Remarque 8. Notez que CBC avec un compteur sera, quant à lui, non sûr du point de vue IND-CPA. En effet l'adversaire peut prédire la valeur du compteur utilisé par le challenger (contrairement à une IV aléatoire), et donc, distinguer les chiffrés à l'aide du premier bloc $E_K(m_0 \oplus IV)$.

CTR avec un compteur

Variable interne : $ctr \leftarrow 0$
$\mathsf{Enc}(K, m)$:

- Séparer m en blocs $m_0, \ldots, m_{t-1}$ de taille n
- Pour tout $i = 0$ à $t-1$: $c_i = m_i \oplus E_K(ctr + i)$
- $ctr \leftarrow ctr + t$
- Renvoyer $ctr - t, c_0, \ldots, c_{t-1}$

$\mathsf{Dec}(K, ctr, c)$:

- Renvoyer $m_i = c_i \oplus E_K(ctr + i)$ pour $i = 0$ à $t-1$

Notez qu'il n'y a pas besoin de remplissage dans le mode CTR, car il suffit de créer une séquence chiffrante assez grande et de ne garder que la longueur dont on a besoin.

Théorème 7. *CTR avec un compteur est IND-CPA si le chiffrement à bloc est une PRP :*

$$\mathrm{Adv}^{IND-CPA}(\mathcal{A}) \leq 2\mathrm{Adv}^{PRP}(\mathcal{B}) + \sigma^2/2^{n+1} \tag{40}$$

où σ est le nombre total de blocs chiffrés dans les requêtes à CTR.

Démonstration. Il y a plusieurs étapes dans la preuve que nous ne détaillerons pas.

- Premièrement, on prouve que lorsqu'on utilise une fonction aléatoire, le mode CTR est IND-CPA. L'avantage de tout adversaire dans ce cas est nul. En effet, lorsque l'adversaire effectue sa requête de chiffrement des messages m_0, m_1, les valeurs du compteur sont de nouvelles valeurs, et le résultat est donc aléatoire et indépendant de toutes les précédentes requêtes de chiffrement.
- Ensuite, on utilise un PRP-PRF switching pour se placer dans le cas où E_K est remplacé par une PRF. Le terme $\sigma^2/2^{n+1}$ vient de là.

On considère maintenant un adversaire $\mathcal{A}$ pour le jeu IND-CPA et on l'utilise pour construire un adversaire $\mathcal{B}$ pour le jeu PRF.

$\mathcal{B}$ a accès à une fonction g et doit déterminer si g est une fonction aléatoire, ou une fonction F_K pour K aléatoire.

$\mathcal{B}$ exécute donc $\mathcal{A}$. Lorsque $\mathcal{A}$ effectue une requête de chiffrement, $\mathcal{B}$ simule cette requête en utilisant son propre compteur et la fonction g à laquelle iel a accès. De plus $\mathcal{B}$ choisit de chiffrer m_0 ou m_1 lors du challenge. $\mathcal{B}$ renvoie 1 si $\mathcal{A}$ réussit, et 0 sinon.

Dans le cas 1 (c'est une fonction F_K), $\mathcal{B}$ renvoie donc 1 avec probabilité :

$$\Pr\left[\mathcal{B} \xrightarrow{PRF} 1\right] = \frac{1}{2} + \frac{1}{2}\mathrm{Adv}^{IND-CPA}(\mathcal{A}) \quad ,$$

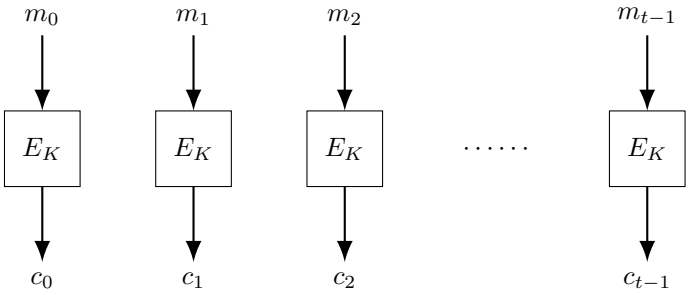


Figure 1: ECB, une insulte à la cryptographie symétrique consistant à chiffrer les blocs indépendamment. Figure de [Jea16].

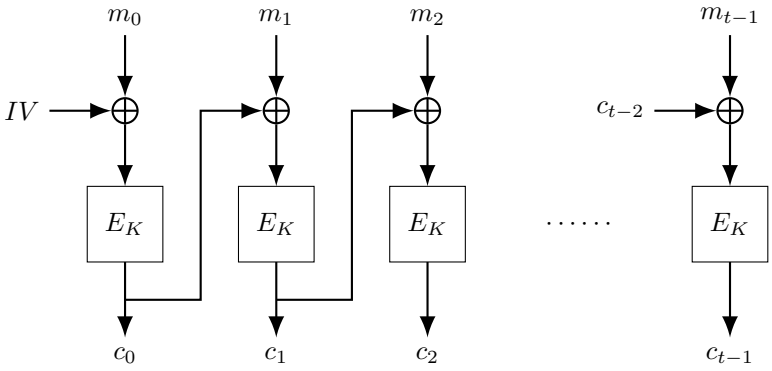


Figure 2: Le mode CBC. Attention à l'IV. Figure de [Jea16].

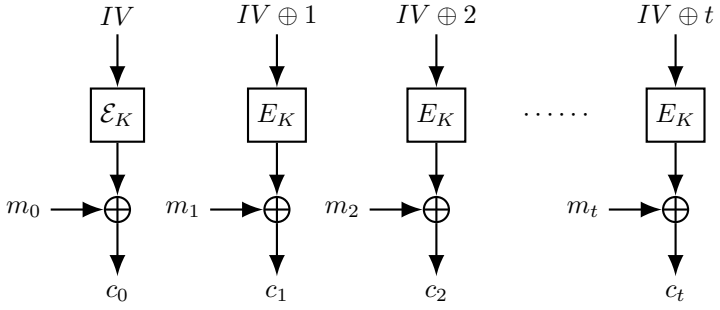


Figure 3: CTR. Figure de [Jea16].

où $\mathrm{Adv}^{IND-CPA}(\mathcal{A})$ renvoie à l'avantage IND-CPA pour attaquer le véritable mode compteur.

Dans le cas 0 (c'est une fonction aléatoire), $\mathcal{B}$ renvoie 1 avec probabilité :

$$\Pr\left[\mathcal{B} \xrightarrow{RAND} 1\right] = \frac{1}{2} + \frac{1}{2}\mathrm{Adv}^{IND-CPA}(\mathcal{A}) \quad ,$$

où $\mathrm{Adv}^{IND-CPA}(\mathcal{A})$ est l'avantage pour attaquer le mode compteur **avec une fonction aléatoire**, qui est nul comme nous l'avons vu. Par conséquent :

$$\mathrm{Adv}^{PRF}(\mathcal{B}) = \Pr\left[\mathcal{B} \xrightarrow{PRF} 1\right] - \Pr\left[\mathcal{B} \xrightarrow{RAND} 1\right] = \frac{1}{2}\mathrm{Adv}^{IND-CPA}(\mathcal{A}) \quad .$$

□

Chiffrement Authentifié. Il existe beaucoup d'autres modes d'opération pour les chiffrement à bloc offrant chacun différents avantages (par exemple CTR est parallélisable, contrairement à CBC) et divers niveaux de sécurité. Cependant, dans la pratique, on ne peut pas seulement utiliser un chiffrement à flot. Il faut aussi **authentifier** pour garantir l'authenticité et l'intégrité du chiffré.

Pour ce faire on utilise soit un mode spécial (*authenticated encryption*) qui fait les deux en un, soit on adjoint au chiffré un MAC (*message authentication code*). En terme de sécurité, un MAC se comporte un peu comme une signature dont la clé de vérification et la clé de signature seraient identiques. Divers modes pour calculer un MAC à l'aide d'un chiffrement à bloc existent. On a ensuite le choix entre différentes compositions (*encrypt-then-MAC*, *MAC-then-encrypt*, etc. [BN08]). La méthode la plus sûre est *encrypt-then-MAC*, où l'on chiffre, on calcule ensuite le MAC du chiffré, et on envoie les deux valeurs.

Exemple 5. Une PRF au sens de la Définition 18, à condition d'être étendue au domaine $\{0,1\}^*$, peut être utilisée comme MAC.

7.4 Fonctions de Hachage

Une fonction de hachage est une fonction qui prend une entrée de taille quelconque et renvoie un **haché** (digest) de taille fixe n :

$$H \ : \{0,1\}^* \to \{0,1\}^n \ . \tag{41}$$

En informatique, on utilise couramment des fonctions de hachage, par exemple dans des tables de hachage. Leur sortie « a l'air aléatoire » de sorte que des **collisions** sont peu fréquentes.

Les fonctions de hachage **cryptographiques** sont différentes des fonctions de hachage générales, car les collisions ne doivent pas être seulement rares : il doit être impossible de pouvoir en exhiber.

Plus formellement, une fonction de hachage requiert trois propriétés fondamentales.

Résistance aux Collisions. Nous avons vu que d'après le paradoxe des anniversaires, on peut trouver une collision d'une fonction $\{0,1\}^* \to \{0,1\}^n$ en temps $\mathcal{O}\left(2^{n/2}\right)$. Une fonction de hachage est résistante aux collisions s'il n'existe pas de méthode plus rapide.

Résistance aux Préimages. On peut toujours inverser une fonction de hachage avec la force brute, en temps $\mathcal{O}(2^n)$. Une fonction de hachage est résistante aux préimages s'il n'existe pas de méthode plus rapide.

Résistance aux Secondes Préimages. Une fonction de hachage est résistante aux secondes préimages s'il n'existe pas de meilleure méthode que la force brute pour résoudre le problème suivant : étant donné $x, H(x) = t$, trouver un autre $y \neq x$ tel que $H(y) = t$.

Remarque 9. Contrairement à la sécurité PRP d'un chiffrement à bloc, ces propriétés ne sont **pas** bien définies d'un point de vue mathématique. En effet, une fonction de hachage est unique. Si nous prenons par exemple la fonction SHA3-256, nous savons qu'il existe des collisions, et il existe donc un algorithme trivial qui renvoie une collision.

Toutefois les notions définies plus haut sont bien établies dans la pratique et ne posent pas de problème.

Exemple : stockage de mots de passe. On ne stocke **pas** des mots de passe en clair dans une base de données. À la place, on stocke le haché du mot de passe (on utilise même pour cela des méthodes spécifiques qui rendent plus dure les attaques par force brute et par dictionnaire). Pour vérifier que le mot de passe est le bon, il suffit de comparer les hachés. En revanche, il est impossible de retrouver le mot de passe à partir de son haché.

7.5 Extension de domaine de Merkle-Dåmgard

Les fonctions de hachage et de chiffrement ont des designs très similaires. Dans les deux cas, on part d'une fonction de taille fixée (chiffrement à bloc ou fonction de compression) et on utilise un mode pour définir une fonction de taille quelconque.

Pour les fonctions de hachage, le mode le plus connu est l'extension de Merkle-Dåmgard

Remplissage. Le **remplissage** (*padding*) est l'opération qui transforme un message de taille quelconque en un multiple de la taille de bloc n. Quand on utilise MD, on a en plus besoin d'encoder la taille du message dans le padding.

Une bonne méthode est donc la suivante. Soit m le message, de longueur $|m|$ en bits. Encoder $|m|$ sous la forme d'un entier ℓ écrit en binaire sur 64 bits. Définir la fonction comme :

$$m \| pad(|m|, n) = m \| 10^* \| \ell \tag{42}$$

où ℓ est un entier de 64 bits et 10^* est un mot avec un seul 1 et des zéros, non vide, de taille telle que la taille de $m\|10^*\|\ell$ est un multiple de n (autrement dit le nombre de 0 est : $n - (|m| + 65) \mod n$.

On peut vérifier que ce remplissage a les propriétés suivantes :

- Le remplissage est inversible ;
- Si deux messages sont de longueurs différentes, alors le bloc de remplissage est différent.

Ces propriétés sont importantes pour la sécurité de MD.

Fonction de compression. Une fonction de compression est simplement une « fonction de hachage » de taille fixe :

$$f \ : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n \ . \tag{43}$$

Il existe différentes manières d'en construire à partir de chiffrements à bloc. Trois modes sont très connus :

- Le mode Davies-Meyer (DM) : $H_i = H_{i-1} \oplus E_{M_i}(H_{i-1})$
- Le mode Matyas-Meyer-Oseas (MMO) : $U_i = M_i \oplus E_{H_{i-1}}(M_i)$
- Le mode Miyaguchi-Preneel (MP) : $H_i = H_{i-1} \oplus M_i \oplus E_{H_{i-1}}(M_i)$

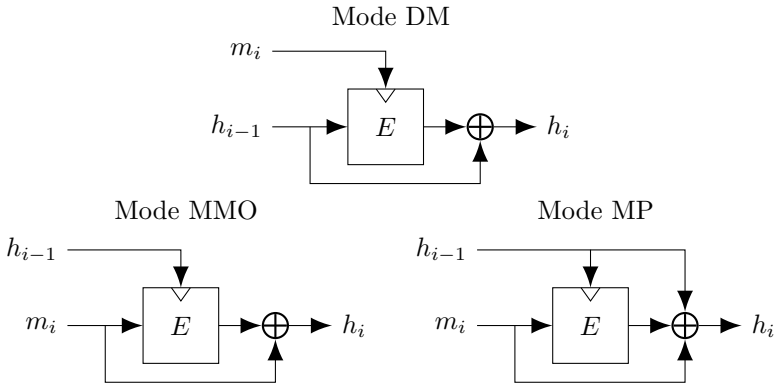


Figure 4: Transformation d'un chiffrement à bloc en fonction de compression. Le triangle correspond à l'entrée de la clé.

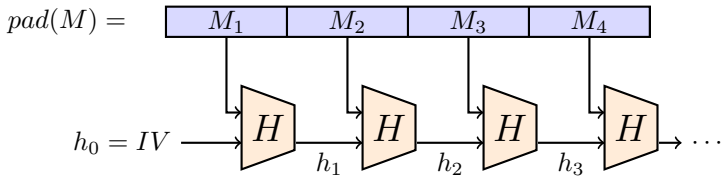


Figure 5: Construction MD

Construction de MD. Une fois que l'on dispose d'une fonction de compression f comme définie plus haut, et d'un schéma de remplissage adapté, la construction MD (figure 5) consiste à absorber les blocs de message l'un après l'autre dans une valeur de chaînage (*chaining value*), dont on renvoie la valeur finale.

Théorème 8 (Informel)**.** *Si la fonction de compression est résistante (aux collisions, préimages, secondes préimages) et qu'on utilise le remplissage défini ci-dessus, alors l'extension MD est résistante.*

7.6 Fonctions Éponges

Si le précédent standard SHA-2 (a.k.a. SHA-256), toujours considéré comme sûr, est basé sur une construction de type Merkle-Dåmgard, la construction **éponge** (*Sponge construction*) est aujourd'hui devenue très populaire. Le standard SHA-3, plus récent standard de hachage du NIST, choisi en 2015 à la suite d'une compétition organisée dans les règles de l'art, est basé sur une éponge (il s'agit de la construction **Keccak**).

Une différence significative par rapport à Merkle-Dåmgard est qu'une éponge est basée sur une **permutation cryptographique**, et non sur une fonction de compression. Le design de la permutation peut ressembler à celui d'un chiffrement à bloc dont on aurait fixé la clé à une valeur publique (ou qui n'aurait pas de clé, tout simplement). La preuve de sécurité du mode éponge suppose que la permutation, comme pour un oracle aléatoire, se comporte comme si on l'avait tirée au hasard parmi toutes les permutations possibles.

L'état interne de l'éponge est divisé en deux parties : la *inner part* de taille c, nommée **capacité**, et la *outer part* de taille r, nommée **rate**. Intuitivement, une valeur de c plus grande augmente la sécurité (qui est typiquement de $c/2$ bits en préimage, seconde préimage, et collision), tandis qu'une valeur de r plus grande augmente la vitesse d'absorption du message, et donc les performances de l'éponge.

Ensuite, le mode éponge se déroule en deux phases : le message M est complété par un padding, et découpé en blocs de taille r. Dans la première phase (**absorption**), les blocs

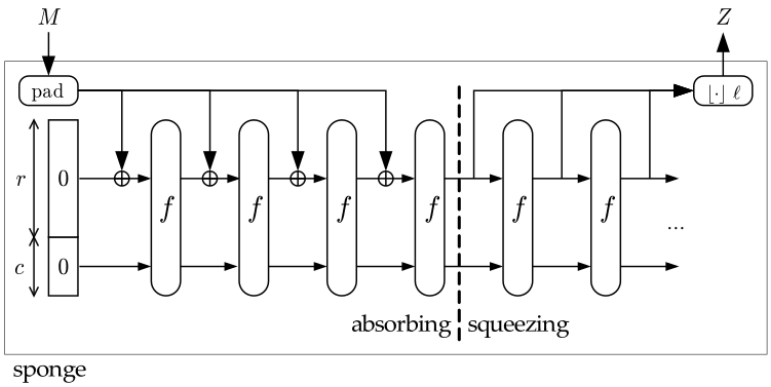


Figure 6: La construction éponge pour une fonction de hachage. Ici f est la permutation. Figure de : https://keccak.team/sponge_duplex.html.

sont XORés à la outer part, et entre deux XORs, on applique la permutation. Dans la deuxième phase (**essorage**), on renvoie la valeur de la *outer part*, une ou plusieurs fois en fonction de la taille de sortie souhaitée, en appliquant toujours la permutation entre deux renvois.

Dans SHA-3, la permutation Keccak a une taille d'état interne de 1600 bits. Le standard SHA-3 définit ensuite plusieurs instances avec une taille de sortie, un rate et une capacité différentes. Par exemple SHA3-224 a une taille de sortie de 224 bits, un rate de 1152 bits et une capacité de 448 bits. Cela correspond à une niveau de sécurité de 112 bits en collision (limité par la taille de sortie) et de 224 bits en préimage (il existe une attaque générique en $2^{c/2}$, qui a donc la même complexité que la recherche exhaustive).

Duplex Sponge. Une modification du mode éponge, la *Duplex Sponge*, permet de construire un mode de chiffrement authentifié avec données associées (AEAD) en utilisant seulement une permutation cryptographique. Lors de la compétition pour la cryptographie légère organisée par le NIST entre 2019 et 2023, de nombreux candidats étaient basés sur des éponges Duplex. Le gagnant ASCON [Dob+21] est basé sur ce mode.

7.7 Conception d'un Chiffrement à Bloc

Toutes les preuves que nous avons vues jusqu'ici ne servent à rien si l'on n'est pas capable d'y placer une primitive efficace et sûre. Nous donnons ici quelques bases de la conception des chiffrements à bloc ; la conception et la cryptanalyse des primitives symétriques sont des domaines de recherche toujours très actifs.

Pour reprendre les termes de Shannon, un chiffrement à bloc : $E_K : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$ doit avoir deux qualités : la **confusion** et la **diffusion**, qui sont définies informellement :

- La diffusion stipule qu'un changement, même mineur, sur le bloc d'entrée va engendrer un changement important et imprévisible du bloc chiffré (il se « diffuse »).
- La confusion stipule que la relation entre le chiffré, le clair et la clé doit être complexe.

Il existe deux grandes méthodes pour construire des chiffrements à bloc (et par extension, des fonctions de compression / permutations cryptographiques). Dans les deux cas, on construit d'abord une **fonction de tour** qui est très simple, et dans laquelle intervient la clé.

- Dans un **réseau de Feistel** (*Feistel network*), l'état interne est séparé en deux parties (L, R) de taille égale (par exemple 32 bits pour le DES). La transformation

de tour a la forme :

$$L, R \mapsto R + f(K, L), L \ ,$$

où f est une fonction (pas nécessairement inversible). Cette structure est utilisée dans le DES.

- Dans un **réseau de substitution-permutation** (SPN), l'état interne est séparé en un ensemble *nibbles*. La fonction de tour comporte une couche de substitution et une couche de permutation que nous allons expliciter plus bas. La clé de tour est typiquement XORée à tout l'état.

Dans les deux cas, comme la taille de clé est proche de la taille de bloc (par exemple 128 ou 256 bits contre 64 ou 128 bits), on définit un algorithme de **cadencement de clé** (*key schedule*) qui prend en entrée la clé maître (*master key*) et renvoie les clés de tour : $K \mapsto k_0, \ldots, k_r$. Un bon *key schedule* est un élément crucial dans la conception d'un chiffrement symétrique.

Le choix du **nombre de tours** crée un compromis entre la sécurité et l'efficacité du chiffrement. Lorsque l'on n'est pas capable de casser le chiffrement entier (ce qui est le but !), on essaie de trouver des attaques sur des versions réduites, avec moins de tours. Le nombre de tours séparant la meilleure attaque de la version complète est appelé **marge de sécurité** (*security margin*).

7.8 Exemple de chiffrement à bloc : AES

Le DES (*data encryption standard*) est un chiffrement à bloc avec 64 bits de bloc et 56 bits de clé, initialement conçu par IBM dans les années 1970, et standardisé ensuite par le *National Bureau of Standards* américain en 1977 sous une version modifiée. La NSA avait notamment rendu le chiffrement plus sûr contre la *cryptanalyse linéaire* et la *cryptanalyse différentielle*, qui étaient encore inconnues à l'époque, et l'avait affaibli contre l'attaque par force brute en réduisant sa taille de clé.

Dans les années 1990, la communauté académique en cryptographie avait gagné en expertise et en assurance, et le DES prenait l'eau de toutes parts : il était évident que sa taille de clé était trop courte, et qu'il était toujours trop faible contre la cryptanalyse linéaire et différentielle.

En 1997, le NIST (*National Institute for Standards and Technology*) américain a donc lancé un processus de standardisation pour trouver le successeur du DES : l'AES (*Advanced Encryption Standard*). Ce processus a pris la forme d'une compétition ouverte dans laquelle des équipes internationales étaient invitées à soumettre leurs chiffrements et à analyser la sécurité des candidats. Cette compétition demeure aujourd'hui une *success story* de la cryptographie académique, qui s'est répétée par la suite avec la compétition SHA-3, la standardisation de chiffrements post-quantiques, etc. Le vainqueur de la compétition était le chiffrement Rijndael, conçu par Joan Daemen et Vincent Rijmen, qui est donc devenu l'AES.

Aujourd'hui, l'AES est le chiffrement le plus utilisé dans le monde, et le standard de 2001 est toujours considéré comme sûr. Les processeurs modernes contiennent même des instructions dédiées (AES-NI) pour exécuter plus rapidement l'algorithme, le rendant particulièrement efficace.

Spécification. L'AES opère sur un bloc de 128 bits avec 128 (10 tours), 192 (12 tours) ou 256 (14 tours) bits de clé. La taille de clé change le nombre de tours et l'algorithme de cadencement de clé, mais pas la fonction de tour en elle-même.

L'état interne de l'AES est une matrice 4×4 d'octets, vus comme desé léments du corps fini $\mathbb{F}_{2^8}$ (donc $16 \times 8 = 128$ bits). Les éléments de $\mathbb{F}_{2^8}$ sont représentés comme des polynômes sur $\mathbb{F}_2$, modulo un polynôme irréductible de degré 8. Dans l'AES on utilise $X^8 + X^4 + X^3 + X + 1$.

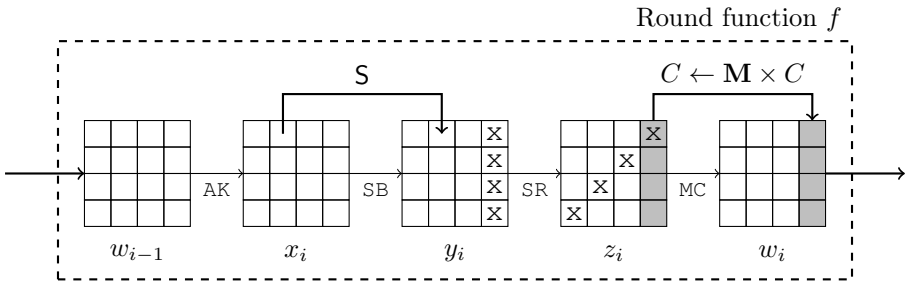


Figure 7: Fonction de tour de l'AES. Figure de [Jea16].

La fonction de tour de l'AES comporte 4 opérations :

- AK : addition de clé de tour
- SB (SubBytes) : application d'une **boîte S** (*S-Box*) aux octets en parallèle
- SR (ShiftRows) : permutation des octets : la ligne i est déplacée de i positions vers la gauche
- MC (MixColumns) : multiplication de chaque colonne (vue comme un vecteur de 4 éléments de $\mathbb{F}_{2^8}$) par la matrice :

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \tag{44}$$

Ici SB est la **couche de substitution** (subtitution layer), qui est aussi la couche non-linéaire du chiffrement. SR et MC forment la **couche de permutation** (permutation layer), qui est la couche linéaire du chiffrement (ici, par « linéaire » , on entend linéaire sur $\mathbb{F}_{2^8}$). Dans la philosophie du SPN, les boîtes S sont des opérations locales mais complexes, qui apportent de la confusion, tandis que les couches linéaires sont de plus grande taille, mais simples, elles apportent surtout de la diffusion.

Exemple 6. Un octet ne diffuse pas à travers SB (les modifications restent sur l'octet), mais diffuse sur 4 octets à travers MC.

La boîte S de l'AES est une permutation sur $\mathbb{F}_{2^8}$. Elle n'a pas été choisie au hasard, mais pour être la plus cryptographiquement sûre possible.

Exemple : cryptanalyse linéaire. Si l'on retire la couche SB de l'AES, le chiffré devient une fonction linéaire (sur $\mathbb{F}_{2^8}$) du plaintext et des clés de tour. À l'aide de paires clair-chiffré, il deviendrait donc possible de retrouver le texte clair en résolvant des équations linéaires.

La **cryptanalyse linéaire** est une généralisation de ce principe dans lequel on utilise une **approximation linéaire** d'une partie du chiffrement, soit une relation de la forme :

$$\text{fonction linéaire de l'entrée} \;=\; \text{fonction linéaire de la sortie}$$

qui serait vraie avec une certaine probabilité, plus grande que l'aléatoire. Une telle propriété distingue le chiffrement d'une PRP et conduit à des attaques de recouvrement de clé. La non-linéarité provient entièrement de la boîte S, et celle de l'AES a été choisie pour être presque optimale.

Exemple 7. Les meilleures attaques de recouvrement de clé sur AES-128 considèrent une version réduite à 7 tours. C'est-à-dire que pour 7 tours, on sait écrire un algorithme retrouvant la clé d'un AES inconnu en moins de 2^{128} calculs. La marge de sécurité actuelle [2] est donc de 3 tours (elle n'a pas bougé depuis dix ans).

2. Dans le contexte d'une PRP, avec une seule clé inconnue, etc.

Résumé

- La cryptographie symétrique part de **primitives** (chiffrement à blox, fonction de compression, permutation) qui sont utilisées dans des **modes** (CTR, CBC, Merkle-Damgård, éponge).
- Les modes CTR et CBC (et d'autres) reposent sur l'hypothèse de sécurité PRP du chiffrement à bloc, qui doit être indistinguable d'une permutation aléatoire.
- La sécurité des primitives repose entièrement sur la cryptanalyse. Un chiffrement moderne comme l'AES est conçu pour être résistant à toutes les formes de cryptanalyse connues. La **marge de sécurité** indique à quel point on est loin d'arriver à casser le chiffrement entier.

Références

[BN08] Mihir Bellare et Chanathip Namprempre. "Authenticated Encryption : Relations among Notions and Analysis of the Generic Composition Paradigm". In : *J. Cryptol.* 21.4 (2008), p. 469-491. DOI : 10.1007/S00145-008-9026-X. URL : https://doi.org/10.1007/s00145-008-9026-x.

[Dob+21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel et Martin Schläffer. "Ascon v1.2 : Lightweight Authenticated Encryption and Hashing". In : *J. Cryptol.* 34.3 (2021), p. 33.

[Jea16] Jérémy Jean. *TikZ for Cryptographers*. https://www.iacr.org/authors/tikz/. 2016.

8 Annexe

8.1 Théorie des Probabilités

Définition 19. Une **distribution de probabilité** P sur un ensemble fini U est une fonction : $P : U \to [0;1]$ telle que $\sum_{x \in U} P(x) = 1$.

Un exemple est la distribution uniforme.

Un sous-ensemble $A \subseteq U$ est appelé un événement. La probabilité de A est définie par : $\Pr[A] = \sum_{x \in A} P(x)$.

Pour deux événements A_1, A_2 : $\Pr[A_1 \cup A_2] \leq \Pr[A_1] + \Pr[A_2]$ avec égalité si $A_1 \cap A_2 = \emptyset$.

Deux événements sont indépendants si $\Pr[A_1 \cap A_2] = \Pr[A_1] \cdot \Pr[A_2]$.

Probabilité conditionnelle :

$$\Pr[A|B] := \frac{\Pr[A \cap B]}{\Pr[B]} \tag{45}$$

Si A et B sont indépendants, $\Pr[A|B] = \prod A$.

Une variable aléatoire X sur V est une fonction...

Théorème 9. *Soit A une variable aléatoire sur $\{0,1\}^n$ et X une variable aléatoire uniforme sur $\{0,1\}^n$, alors $Y = A \oplus X$ est une variable aléatoire uniforme sur $\{0,1\}^n$.*

Démonstration. Pour $n = 1$.

$$\Pr[Y = 0] = \Pr[A = 0 \wedge X = 0] + \Pr[A = 1 \wedge X = 1] = \frac{1}{2}\Pr[A = 0] + \frac{1}{2}\Pr[A = 1] = \frac{1}{2} \ .$$

□

8.2 Théorie des Nombres

Premiers et Divisibilité. Soit $\mathbb{Z}$ l'anneau des entiers. Pour $a, b \in \mathbb{Z}$ nous disons que a divise b ($a|b$) s'il existe $k \in \mathbb{Z}$ tel que $b = ka$. Si $a \notin \{1, b\}$, a est appelé un **facteur** de b. Un entier p est premier s'il n'a aucun facteur. Tout entier se décompose en un unique produit de facteurs premiers.

Pour deux entiers a, b, le plus grand c tel que $c|a$ et $c|b$, noté $gcd(a, b)$, est calculable en temps polynomial par l'algorithme d'Euclide étendu. Il permet aussi de calculer en temps polynomial des entiers x, y tels que $ax + by = c$.

Arithmétique modulaire. Soit $a, b, N \in \mathbb{Z}$ avec $N > 1$. La notation $a \pmod N$ signifie le reste dans la division euclidienne par N. On note $a = b \pmod N$ pour $a \pmod N = b \pmod N$ et on dit que a et b sont congrus modulo N.

La relation de congruence modulo N est une relation d'équivalence, qui satisfait les propriétés standard de l'arithmétique au regard de l'addition, soustraction, multiplication.

S'il existe a tel que $ab = 1 \pmod N$ on dit que b est l'inverse de a modulo N, noté a^{-1}. a est inversible modulo N si et seulement si $gcd(a, N) = 1$.

Opérations dans $\mathbb{Z}_N$. On note $\mathbb{Z}_N$ pour $\mathbb{Z}/N\mathbb{Z}$, le groupe des entiers modulo N. L'addition, la multiplication des entiers sont des opérations efficaces.

Lorsqu'un entier a est premier avec N, il admet un inverse multiplicatif modulo N : b tel que $ab = \mod N$. Cet inverse peut se calculer efficacement à l'aide de l'algorithme d'Euclide étendu.

L'ensemble $\mathbb{Z}_N^*$ est l'ensemble des éléments inversibles de $\mathbb{Z}_N$, qui est donc l'ensemble des entiers $1 \leq x \leq N - 1$ premiers avec N. Il forme un groupe multiplicatif. Lorsque N est premier, ce groupe est d'ordre $N - 1$.

Groupes. Nous étudions des groupes finis, i.e., un ensemble muni d'une loi • associative ; • avec un élément neutre (souvent noté 1) ; • où tout élément a un inverse. De plus nous ne considérons que des groupes abéliens (la loi est commutative).

Dans un groupe, l'exponentiation $g, x \mapsto g^x$ peut être calculée avec un nombre d'opérations polynomial en la taille de x.

Groupe $\mathbb{Z}_N^*$. Pour tout $N > 1$, l'ensemble $\mathbb{Z}_N = \{0, \ldots, N-1\}$ muni de l'addition modulo N est un groupe. On définit $\mathbb{Z}_N^* = \{a \in \{1, \ldots, N-1\}, gcd(a, N) = 1\}$ l'ensemble des entiers premiers avec N dans $\mathbb{Z}_N$. Muni de la multiplication modulo N, c'est un groupe.

Indicatrice d'Euler. L'indicatrice d'Euler ϕ est la fonction telle que $\phi(N)$ est le nombre d'entiers dans $\mathbb{Z}_N$ qui sont premiers avec N (qui donne donc l'ordre de $\mathbb{Z}_N^*$).

Si N est premier on a donc : $\phi(N) = N - 1$.

Si N se factorise sous la forme :

$$N = \prod_{i=1}^{\ell} p_i^{e_i} \tag{46}$$

alors

$$\phi(N) = \prod_{i=1}^{\ell} p_i^{e_i - 1}(p_i - 1) \ . \tag{47}$$

En particulier si $N = PQ$ est le produit de deux nombres premiers distincts, $\phi(N) = (P-1)(Q-1)$.

Théorème 10 (Théorème de Lagrange)**.** *Dans un groupe $(\mathbf{G}, \cdot)$ d'ordre n, pour tout $a \in \mathbf{G}$, on a $a^n = 1$.*

On en déduit que si $x \in \mathbb{Z}_N^*$ alors $x^{\phi(N)} = 1 \mod N$. Le cas où N est premier et $\phi(N) = N - 1$ est le petit théorème de Fermat.

CRT. Le théorème des restes Chinois (CRT) établit un isomorphisme de groupes.

Théorème 11. *Soit $N = PQ$ où P, Q sont premiers entre eux :*

$$\Big\{ \mathbb{Z}_N \simeq \mathbb{Z}_P \times \mathbb{Z}_Q \mathbb{Z}_N^* \simeq \mathbb{Z}_P^* \times \mathbb{Z}_Q^* \tag{48}$$

La fonction $f(x) = (x \mod P, x \mod Q)$ est un tel isomorphisme.

On peut étendre ce théorème à produit de ℓ entiers premiers deux à deux.

Lorsqu'on connaît la factorisation de N, on peut convertir d'un groupe à l'autre en temps polynomial. Dans un sens, c'est facile. Dans l'autre, on utilise l'algorithme d'Euclide étendu pour calculer x, y tels que $xP + yQ = 1$.

Alors étant donné $(a, b) \in \mathbb{Z}_P \times \mathbb{Z}_Q$, on calcule $c = yQa + xPb \pmod N$. On vérifie que $c \pmod P = yQa \pmod P = a$ et que $c \pmod Q = xPb \pmod Q = b$.

Groupes Cycliques. Tout groupe d'ordre premier p est cyclique, c'est-à-dire qu'il existe un élément g tel que $\mathbf{G} = \{1, g, g^2, \ldots, g^{p-1}\}$ (cela vient du théorème de Lagrange).

Un autre exemple important :

Théorème 12. *Si p est premier, alors $\mathbb{Z}_p^*$ est cyclique.*