



Notes de cryptographie

Roberto M. Amadio

► To cite this version:

| Roberto M. Amadio. Notes de cryptographie. École d'ingénieur. France. 2018. cel-01957471v1

HAL Id: cel-01957471

<https://hal.archives-ouvertes.fr/cel-01957471v1>

Submitted on 17 Dec 2018 (v1), last revised 3 Jan 2022 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Notes de cryptographie

Roberto M. Amadio
Université Paris-Diderot

17 décembre 2018

Table des matières

| | |
|---|-----------|
| Préface | 7 |
| Notation | 9 |
| 1 Introduction | 11 |
| 1.1 Une première définition de système cryptographique | 11 |
| 1.2 Systèmes symétriques et asymétriques | 13 |
| 1.3 Mode dictionnaire (ECB) | 14 |
| 1.4 Exemples de systèmes mono-alphabétiques | 14 |
| 1.5 Systèmes affines (en dimension n) | 16 |
| 1.6 Un exemple de protocole cryptographique | 19 |
| 1.7 Sommaire | 20 |
| 2 Confidentialité parfaite | 21 |
| 2.1 Système à masque jetable | 21 |
| 2.2 Confidentialité parfaite | 22 |
| 2.3 Caractérisation de la confidentialité parfaite | 23 |
| 2.4 Sommaire | 24 |
| 3 Distance statistique et équivalence calculatoire | 25 |
| 3.1 Distance statistique | 25 |
| 3.2 Temps probabiliste polynomial | 28 |
| 3.3 Fonctions négligeables | 30 |
| 3.4 Équivalence calculatoire | 30 |
| 3.5 Sommaire | 32 |
| 4 Générateurs pseudo-aléatoires de nombres (PRG) et 1-confidentialité | 33 |
| 4.1 PRG | 33 |
| 4.2 1-Confidentialité | 34 |
| 4.3 PRG : théorie et pratique | 36 |
| 4.4 Sommaire | 37 |
| 5 Générateurs pseudo-aléatoires de fonctions (PRF) et CPA-confidentialité | 39 |
| 5.1 Fonctions (pseudo-)aléatoires | 39 |
| 5.2 CPA-Confidentialité | 41 |
| 5.3 PRF et mode compteur (CTR) | 43 |
| 5.4 Mode chaîné (CBC) | 44 |
| 5.5 Confidentialité CPA en mode CBC | 44 |
| 5.6 Schéma de Feistel | 45 |
| 5.7 Sommaire | 46 |
| 6 Intégrité | 47 |
| 6.1 Codes d'authentification de messages (MAC) | 47 |
| 6.2 CBC-MAC | 48 |
| 6.3 Combiner confidentialité et intégrité | 49 |
| 6.4 Sommaire | 50 |

| | | |
|-----------|--|-----------|
| 7 | Permutations pseudo-aléatoires : pratique | 51 |
| 7.1 | DES | 51 |
| 7.2 | Un exemple de réseau de substitution permutation | 54 |
| 7.3 | Structure d'AES | 55 |
| 7.4 | Sommaire | 56 |
| 8 | Systèmes asymétriques | 59 |
| 8.1 | CPA confidentialité : cas asymétrique | 59 |
| 8.2 | Système RSA | 60 |
| 8.3 | Échange de clefs : protocole de Diffie-Hellman | 62 |
| 8.4 | Système d'ElGamal | 63 |
| 8.5 | Sommaire | 64 |
| 9 | Hachage et compression | 65 |
| 9.1 | Paradoxe des anniversaires | 66 |
| 9.2 | Construction de Merkle-Damgård | 66 |
| 9.3 | Compression de Davies-Meyer | 68 |
| 9.4 | Compression CHP | 68 |
| 9.5 | Sommaire | 70 |
| 10 | Intégrité : cas asymétrique | 71 |
| 10.1 | Signature RSA | 71 |
| 10.2 | Sommaire | 72 |
| A | Arithmétique | 75 |
| A.1 | Division et reste | 75 |
| A.2 | Plus grand commun diviseur (pgcd) | 76 |
| A.3 | Nombres premiers | 79 |
| A.4 | Congruences et structure des entiers modulo | 80 |
| A.5 | Exponentiation rapide | 81 |
| A.6 | Restes chinois | 81 |
| A.7 | Groupe multiplicatif | 83 |
| A.8 | Test de primalité | 85 |
| A.9 | Groupes cycliques et générateurs | 86 |
| B | Algèbre linéaire | 89 |
| C | Travaux pratiques et dirigés | 91 |
| C.1 | Utilisation de Gnu Privacy Guard | 91 |
| C.2 | Cryptoanalyse de chiffrements affines | 91 |
| C.3 | Vernam faible | 92 |
| C.4 | Mise en gage d'un bit | 93 |
| C.5 | Réseaux de substitution permutation | 93 |
| C.6 | Programmation de RSA en Java | 95 |
| C.7 | Routage anonyme | 95 |
| C.8 | Pseudo-primalité | 96 |
| C.9 | Calcul du logarithme discret | 97 |
| C.10 | Signature aveugle | 97 |
| D | Exercices | 99 |
| D.1 | Générateur affine | 99 |
| D.2 | Combinaison de systèmes cryptographiques | 99 |
| D.3 | Permutation et Vigenère jetable | 100 |
| D.4 | Distance statistique et équivalence calculatoire | 100 |
| D.5 | PRG à sens unique | 100 |
| D.6 | Tolérance aux erreurs du mode CBC | 100 |
| D.7 | Feistel faible | 100 |
| D.8 | Feistel et PRP | 101 |

| | |
|--|------------|
| D.9 Mac faibles | 101 |
| D.10 Substitution non-linéaire | 101 |
| D.11 RSA faible | 102 |
| D.12 Diffie-Hellman en groupe | 102 |
| D.13 Programmation fonction de hachage en Java | 102 |
| D.14 Signature RSA à anneau | 102 |
| D.15 Test pour les résidus quadratiques | 103 |
| D.16 Entiers de Blum | 103 |
| Bibliographie | 105 |
| Index | 107 |

Préface

On trouve de nombreux livres, notes de cours, vidéos, . . . qui proposent une introduction adéquate à la cryptographie. Dans ce sens ces notes de cours sont redondantes ; elles n'ont d'autre ambition que de fournir une *trace synthétique* des sujets traités dans un cours d'initiation à la cryptographie pour les étudiants de l'Université Paris-Diderot. En particulier, la lecture du livre [KL07], ou d'un autre texte d'un niveau équivalent, est fortement conseillée. Le livre en préparation [BS16] est aussi un complément intéressant.

L'objectif principal du cours est d'introduire les méthodes pour *définir* les notions de confidentialité et d'intégrité, et pour *construire* des fonctions cryptographiques qui assurent ces propriétés sous certaines hypothèses bien identifiées.

Ces notes s'appuient sur un certain nombre de notions mathématiques qui sont normalement couvertes dans un premier cycle scientifique. On suppose notamment des notions élémentaires de théorie des groupes, d'algèbre linéaire, d'arithmétique modulaire et de calcul des probabilités. Les faits essentiels sont rappelés en appendice et le livre [Sho05] permet d'approfondir les sujets évoqués. Ces notes sont complétées par un certain nombre de travaux pratiques et dirigés.

Du côté informatique, on suppose que le lecteur est à l'aise avec la programmation dans un langage de haut niveau et qu'il maîtrise les notions de base de l'algorithmique (structures de données, complexité asymptotique, . . .).

La cryptographie n'est qu'une petite partie de ce qu'on appelle la *sécurité informatique*. Le livre [And01] donne une vue d'ensemble de ce vaste domaine. Certains soutiennent que la cryptographie est la partie de la sécurité informatique dans laquelle on fait des preuves. D'autres vous diront qu'on n'attaque pas les systèmes cryptographiques mais qu'on les contourne. Le lecteur devrait garder à l'esprit ces deux points de vue.

Plus en général, la cryptographie a un impact sur l'organisation économique, politique et militaire du monde. En retour, l'économie, la politique et les organisations militaires motivent et polluent la recherche en cryptographie. Sur ce sujet, une lecture intéressante est [Rog15].

Notation

Ensembles

| | |
|------------------------|--|
| \emptyset | ensemble vide |
| \mathbf{N} | nombres naturels |
| \mathbf{Z} | nombres entiers |
| \mathbf{Q} | nombres rationnels |
| \mathbf{R} | nombres réels |
| \cup, \cap | union, intersection de deux ensembles |
| \bigcup, \bigcap | union, intersection d'une famille d'ensembles |
| X^c | complémentaire de X |
| $\mathcal{P}(X)$ | sous-ensembles de X |
| $\mathcal{P}_{fin}(X)$ | sous-ensembles finis de X |
| Y^X | fonctions de X dans Y |
| $\sharp X$ | cardinal de X |
| R^* | clôture réflexive et transitive d'une relation binaire R |

Arithmétique

| | |
|------------------|---|
| \mathbf{Z}_n | entiers modulo n |
| \mathbf{Z}_n^* | groupe multiplicatif des entiers modulo n |
| $ a $ | valeur absolue |
| \equiv | congruence |
| a/b | quotient division entière |
| $a \bmod b$ | reste de la division entière (ou module) |
| $a \mid b$ | divisibilité |

Algèbre linéaire

| | |
|---------------|------------------|
| A, B, \dots | matrices |
| $adj(A)$ | matrice adjointe |
| $det(A)$ | déterminant |
| A^{-1} | matrice inverse |

Probabilité

| | |
|------------------------------|--|
| Ω | ensemble des expériences |
| \mathcal{A} | ensemble des événements |
| $P(A)$ | probabilité d'un événement |
| $P(A \mid B)$ | probabilité conditionnelle |
| X, Y | variables aléatoires discrètes (v.a.d) |
| $\Delta(X, Y)$ | distance statistique |
| $\{X_n\}_{n \in \mathbf{N}}$ | famille de v.a.d. (<i>ensemble</i>) |
| \sim | équivalence calculatoire de familles de v.a.d. |

Complexité asymptotique

| | |
|-----------------|---|
| f est $O(g)$ | $\exists n_0, k \geq 0 \forall n \geq n_0 (f(n) \leq k \cdot g(n))$ |
| f polynomiale | $\exists d \geq 0 f$ est $O(n^d)$ |
| f négligeable | $\forall d \geq 0 f$ est $O(n^{-d})$ |

Cryptographie

| | |
|---------------|--|
| \mathcal{P} | textes clairs |
| \mathcal{C} | textes chiffrés |
| \mathcal{K} | clefs |
| (G, E, F) | système cryptographique (chiffrement) |
| (G, Mac, V) | système d'authentification (MAC) |
| \oplus | ou exclusif |
| \mathcal{G} | générateur pseudo-aléatoire de nombres (PRG) |
| \mathcal{F} | générateur pseudo-aléatoire de fonctions (PRF) |

Chapitre 1

Introduction

La *crypto-logie* se décompose en deux activités : la *crypto-graphie* qui est l'art de chiffrer les communications et la *crypto-analyse* qui est l'art d'analyser les communications chiffrées. En pratique, dans ces notes on utilise le terme *cryptographie* pour se référer aux deux activités.

La cryptographie a une longue histoire pimentée par beaucoup d'anecdotes amusantes (lire par exemple [Sin99]). Pendant longtemps elle a été traitée comme une activité mystérieuse mais un changement radical est intervenu autour de 1980 avec la diffusion des ordinateurs, l'intérêt commercial et une certaine consolidation mathématique (système DES, système RSA, liaison avec la théorie de la complexité, ...) Techniquement, il s'agit d'une discipline à l'intersection de différents domaines scientifiques comme la théorie des nombres, le calcul des probabilités, la théorie de l'information et du codage et la théorie de la complexité du calcul.

Dans ce chapitre, on présente un certain nombre de systèmes cryptographiques qui ont une valeur historique et qui servent comme introduction au sujet. Tous ces systèmes ont des défauts importants qui les rendent inutilisables en pratique mais l'analyse de ces défauts nous permettra d'apprécier les difficultés du sujet et de raffiner nos méthodes.

1.1 Une première définition de système cryptographique

A quoi sert un système cryptographique ? Une première tâche élémentaire consiste à assurer la *confidentialité* d'une communication.

- *A* souhaite envoyer un *message* à *B* sur un *canal* où un attaquant *C* peut *écouter* (*eavesdrop* en anglais).
- *A* et *B* se mettent d'accord sur une méthode pour *chiffrer* et *déchiffrer* le message.
- *A* *chiffre* le message clair avant de l'envoyer.
- *B* *déchiffre* le message chiffré après l'avoir reçu.

On peut déjà distinguer une première question : comment définir le fait que l'attaquant *C*, qui a accès au message chiffré, n'obtient pas d'information significative sur le contenu du message clair ?

Par ailleurs, on peut envisager nombreuses variations. En effet l'hypothèse *minimale* est que l'attaquant *C* a accès au texte chiffré (*texte chiffré connu*). Un attaquant plus puissant peut avoir accès à un certain nombre de couples composés de textes clairs et de leurs chiffrements (*texte clair connu*). Par exemple, l'attaquant peut savoir que tout message chiffré commence d'une certaine façon. Un attaquant encore plus puissant peut choisir les textes clairs dont connaître le chiffrement (*texte clair choisi*) et/ou les textes chiffrés dont connaître

le déchiffrement (*texte chiffré choisi*). Par exemple, l'attaquant arrive à confondre un participant honnête pour qu'il fournisse en bonne foi les chiffréments et/ou les déchiffrements souhaités.

L'attaquant C pourrait être *actif* dans le sens que non seulement il peut lire tous les messages qui passent sur un canal mais il peut aussi en effacer et en générer certains. En présence d'un attaquant actif, il convient de se soucier non seulement de la *confidentialité* des messages échangés mais aussi de leur *intégrité*. Le récepteur B du message doit avoir une méthode pour s'assurer que le message a bien été émis par A et qu'il ne s'agit pas, par exemple, d'un message produit par l'attaquant C .

La confidentialité et l'intégrité sont les deux propriétés fondamentales qui doivent être assurées par un système cryptographique. Cependant, dans certaines situations d'autres propriétés apparaissent. Dans un système de vote électronique, on souhaite garantir une forme d'*anonymat* : l'observation du vote ne devrait pas permettre de révéler comment un participant a voté. Dans un système de commerce électronique, le *vendeur* doit avoir une façon de prouver que l'*acheteur* a bien passé une commande ; on parle de propriété de *non-répudiation*.

On introduit maintenant une première définition de système cryptographique qui sera raffinée dans les chapitres suivants. On se focalisera en particulier sur des systèmes cryptographiques qui visent à assurer la *confidentialité*. On verra dans le chapitre 6, que la méthodologie et les outils développés s'adaptent assez facilement au problème de l'*intégrité*.

Définition 1 (système cryptographique) *Un système cryptographique est composé d'un vecteur $(\mathcal{P}, \mathcal{C}, \mathcal{K}, G, E, D)$ où :*

- $\mathcal{P}, \mathcal{C}, \mathcal{K}$ sont des ensembles finis (non-vides) qui représentent, respectivement, l'ensemble des textes clairs, des textes chiffrés et des clefs.
- G, E, D sont des fonctions calculables de type :

$$G : 1 \rightarrow \mathcal{K} , \quad E : \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C} , \quad D : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{P} ,$$

qui représentent respectivement la génération d'une clef, le chiffrage d'un texte clair avec une clef et le déchiffrement d'un texte chiffré avec une clef. En écrivant E_k pour $E(-, k)$ et D_k pour $D(-, k)$, on suppose :

$$\forall k_1 \in \mathcal{K} \quad \exists k_2 \in \mathcal{K} \quad \forall p \in \mathcal{P} \quad D_{k_2}(E_{k_1}(p)) = p$$

En d'autres termes, toute fonction de chiffrage E_{k_1} a une inverse à gauche D_{k_2} .

- Remarque 2**
1. *L'existence d'une inverse à gauche de toute fonction de chiffrage E_k implique que la fonction E_k doit être injective. Si l'on fixe un texte chiffré et une clef il y a donc au plus un texte clair qui correspond au texte chiffré.*
 2. *On suppose que tous les ensembles sont finis. Pour chiffrer un message de longueur arbitraire il faudra donc concevoir une méthode pour le découper en morceaux et le chiffrer.*
 3. *Les fonctions de génération de clef G et de chiffrage E peuvent être probabilistes ce qui veut dire que ces fonctions ont accès à un générateur de bits aléatoires et qu'elles peuvent s'en servir pour manipuler leur flot de contrôle. Dans les systèmes pratiques considérés dans ces notes les fonctions G et E seront toujours probabilistes.*

4. Les algorithmes de chiffrement et de déchiffrement sont publics. La sécurité du système repose seulement sur le fait qu'une certaine clef n'est pas connue par l'attaquant.
5. La définition est clairement incomplète car elle n'impose aucun standard de sécurité. Par exemple, d'après cette définition on pourrait toujours prendre $\mathcal{P} = \mathcal{C}$ et supposer que les fonctions E_k et D_k sont les fonctions identité.

1.2 Systèmes symétriques et asymétriques

On distingue deux types principaux de systèmes cryptographiques : les systèmes symétriques (ou à clef privée) et les systèmes asymétriques (ou à clef publique).

Dans les systèmes symétriques il est facile de calculer la clef de déchiffrement à partir de la clef de chiffrement (et vice versa). Leur utilisation typique est la suivante :

1. A génère de façon aléatoire une nouvelle clef k .
2. A communique la nouvelle clef k à B en utilisant un canal sûr.
3. A et B échangent des messages en utilisant la clef k .

Parmi les systèmes symétriques utilisés actuellement on peut citer ceux basés sur les fonctions DES et AES [DR02]. En général, ces systèmes permettent un chiffrement et déchiffrement rapide et ils assurent une sécurité convenable avec des 'petites' clefs (de l'ordre de 10^2 bits). Tous les systèmes historiques sont des systèmes symétriques car l'idée de système asymétrique est apparue seulement autour de 1980. Un inconvénient de ces systèmes est qu'un canal sûr est nécessaire pour initialiser la communication. Par ailleurs, pour chaque correspondant, il faut maintenir une clef secrète.

Dans les systèmes asymétriques, les clefs de chiffrement et déchiffrement sont différentes et il est *difficile* de calculer l'une à partir de l'autre. Leur utilisation typique est la suivante :

1. B génère un couple de clefs (k_1, k_2) telles que $D_{k_2} \circ E_{k_1} = id_{\mathcal{P}}$.
2. B rend k_1 publique et maintient k_2 secrète.
3. A peut apprendre la clef publique k_1 , chiffrer des messages avec k_1 et les envoyer à B . Dans ce cas, seulement B sera capable de déchiffrer ces messages en utilisant la clef secrète k_2 .

Parmi les exemples de systèmes asymétriques, on peut citer le système *RSA* [RSA78] et le système d'El Gamal [Gam85]. Un avantage des systèmes asymétriques est qu'il ne faut pas se soucier de garder secrète la clef de chaque correspondant (la clef est publique). Néanmoins, il faut s'assurer de l'authenticité de la clef utilisée. Ainsi si A veut envoyer un message à B il doit être sûr d'utiliser la clef publique de B . Un inconvénient des systèmes asymétriques par rapport aux systèmes symétriques est que souvent ils demandent des calculs plus compliqués et une clef plus grande (de l'ordre de 10^3 bits). Cependant, on peut combiner un système asymétrique avec un système symétrique (on parle de chiffrement *hybride*) de façon à avoir la flexibilité du système asymétrique combinée avec l'efficacité du système symétrique. Aussi des systèmes asymétriques introduits plus récemment qui sont basés par exemple sur les courbes elliptiques, les réseaux de nombres, ... permettent de réduire la taille des clefs.

On retrouve la distinction entre symétrique et asymétrique aussi dans les systèmes conçus pour assurer l'*intégrité*. L'idée générale est d'attacher au message transmis une *étiquette* qui permet de vérifier à la fois l'*origine* du message et son *intégrité*.

- Dans les systèmes symétriques, l'étiquette est générée à l'aide d'une clef secrète qui est partagée par l'émetteur et le récepteur du message. On parle de *codes d'authentification de messages* (en anglais, *message authentication codes ou MAC*).
- Dans les systèmes asymétriques, l'émetteur a un couple de clefs dont l'une est publique et l'autre est privée. L'émetteur se sert de la clef privée pour produire l'étiquette et tout récepteur du message peut utiliser la clef publique pour vérifier son intégrité. On parle de *signatures digitales*.

1.3 Mode dictionnaire (ECB)

Dans la suite du chapitre, on va présenter un certain nombre de systèmes cryptographiques historiques. Par défaut, on fera l'hypothèse qu'un texte clair est un mot sur l'alphabet français (et anglais) $\Sigma = \{A, B, \dots, Z\}$; ou de façon équivalente, un mot sur \mathbf{Z}_{26} (les entiers modulo 26). Par ce choix, on fait l'hypothèse que : espaces, ponctuations, lettres minuscules et majuscules sont ignorées. Ainsi *I wake up early* devient *IWAKEUPEARLY*. Ceci n'est pas très gênant tant que le texte clair correspond à un texte en langue naturelle car les langues naturelles sont très redondantes et il est assez facile de retrouver les espaces dans le texte.

Dans les systèmes présentés l'espace des texte clairs et des texte chiffrés est constitué de mots sur l'alphabet Σ de longueur n :

$$\mathcal{P} = \mathcal{C} = \Sigma^n \quad n \geq 1 .$$

On ne traite que des systèmes symétriques et les fonctions de chiffrement et de déchiffrement sont des *permutations* sur Σ^n . Pour chiffrer un texte clair composé de m caractères, on applique un *bourrage* pour obtenir un texte clair de longueur $m' \geq m$ où $m' = \ell \cdot n$ est un multiple de n . On obtient donc un message $M = M_1 \cdots M_\ell$ composé de ℓ blocs de n caractères que l'on chiffre en appliquant la fonction E_k bloc par bloc :

$$E_k(M) = E_k(M_1) \cdots E_k(M_\ell) .$$

De façon similaire, le déchiffrement d'un texte composé de ℓ blocs de n caractères consiste à appliquer la fonction D_k bloc par bloc et à éliminer le bourrage. Dans ce contexte, on dit qu'on chiffre les blocs en *mode ECB* (pour *electronic codebook mode*). En français, on parle aussi de *mode dictionnaire* car à chaque bloc de texte clair correspond un bloc de texte chiffré.¹

En l'absence d'autres informations sur le texte clair, le bourrage peut se faire de la façon suivante : si m n'est pas un multiple de n on complète le dernier bloc avec les caractères $XY \cdots Y$ et si m est un multiple de n alors on ajoute un bloc avec les caractères $XY \cdots Y$ (le choix des caractères X et Y est arbitraire). Ainsi après déchiffrement du message on obtient un texte qui termine avec un mot de la forme $XY \cdots Y$ que l'on peut ignorer car il ne fait pas partie du message initiale.²

1.4 Exemples de systèmes mono-alphabétiques

On donne des exemples de systèmes historiques dans lesquels le chiffrement se fait caractère par caractère d'après une fonction sur l'alphabet Σ . On appelle ces systèmes mono-

1. D'autres modes de transmission seront abordés dans la suite du cours, notamment le mode chaîné (*CBC*) et le mode compteur (*CTR*).

2. Ce système de bourrage permet une attaque sur texte clair connu, mais de toute façon les systèmes historiques que l'on présente ont d'autres faiblesses...

alphabétiques : un caractère est toujours chiffré de la même façon, indépendamment de sa position dans le texte.

Système de César

Dans ce système un bloc est composé d'un caractère et une clef est aussi un caractère. Ainsi on a :

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbf{Z}_{26} .$$

Pour une clef $k \in \mathbf{Z}_{26}$ on définit :

$$E_k(p) = (p + k) \bmod 26, \quad D_k(c) = (c - k) \bmod 26 .$$

En appliquant les propriétés de l'arithmétique modulaire on vérifie :

$$\begin{aligned} D_k(E_k(p)) &= ((p + k) \bmod 26 - k) \bmod 26 \\ &= (p + k - k) \bmod 26 \\ &= p . \end{aligned}$$

On suppose que l'attaquant sait quand un texte suffisamment long est un texte significatif (par exemple, il sait que le texte clair est rédigé dans une certaine langue naturelle). Dans ce cas, il peut monter une attaque par énumération sur le texte chiffré. Étant donné un message chiffré $c = c_1 \dots c_n$ il essaye les 26 clefs $k \in \mathbf{Z}_{26}$ et il calcule :

$$(c_1 - k) \bmod 26 \dots (c_n - k) \bmod 26 .$$

La morale de cette attaque est que dans un système cryptographique l'espace des clefs doit être assez grand.

Système par substitution

On élargit l'espace des clefs. Soient :

$$\mathcal{P} = \mathcal{C} = \mathbf{Z}_{26} , \quad \mathcal{K} = \{ \pi : \mathbf{Z}_{26} \rightarrow \mathbf{Z}_{26} \mid \pi \text{ permutation} \} .$$

Étant donnée une permutation π , on définit :

$$E_\pi(p) = \pi(p) , \quad D_\pi(c) = \pi^{-1}(c) .$$

On note qu'à partir de π il est facile de calculer la permutation inverse π^{-1} . On vérifie que :

$$D_\pi(E_\pi(p)) = (\pi^{-1} \circ \pi)(p) = p .$$

On a donc un système cryptographique symétrique avec $26!$ clefs possibles et une attaque par énumération semble difficile. Cependant, la fréquence des lettres dans les langages naturels a été bien étudiée. Par exemple, la table 1.1 donne la fréquence des lettres les plus communes en anglais et en français.

Étant donné un texte suffisamment long qui est le chiffrement d'un texte clair en anglais ou en français, la lettre qui est présente le plus souvent correspond probablement à e , et ainsi de suite... Comme les langages naturels sont redondants, des qu'un nombre limité de substitutions ont été déterminées, il est possible de deviner celles qui restent. Par ailleurs, il y a des statistiques aussi sur la fréquence de digrammes et de tri-grammes. Par exemple, THE est un tri-gramme qu'on trouve souvent en anglais.

| Lettre | Anglais | Français |
|--------|---------|----------|
| e | 12,7 | 17,8 |
| a | 8,2 | 8,2 |
| o | 7,5 | 5,7 |
| i | 7,0 | 7,3 |
| t | 9,1 | 7,3 |
| s | 6,3 | 8,3 |

TABLE 1.1 – Fréquence des lettres les plus communes en anglais et en français

Système affine (en dimension 1)

Dans le système de César, on a considéré les caractères comme des entiers modulo 26 et on a vu le chiffrement comme une fonction de la forme $x \mapsto (x + k) \bmod 26$. On va maintenant considérer des transformations affines de la forme :

$$x \mapsto (ax + b) \bmod 26 .$$

On sait que cette transformation est inversible si et seulement si $\text{pgcd}(a, 26) = 1$ (voir proposition 91). Il en suit qu'on a $12 \cdot 26 = 312$ choix possibles pour la clef $(a, b) \in \mathbf{Z}_{26}^* \times \mathbf{Z}_{26}$. Il s'agit d'un petit nombre et donc ce système est aussi sujet à une attaque par énumération. Par ailleurs, il y a une deuxième faiblesse : la fonction de chiffrement est affine et ce fait permet de monter une attaque sur texte clair connu comme illustré dans l'exercice suivant.

Exercice 3 *On sait qu'un système affine avec clef (a, b) envoie E sur R et S sur H . Dérivez de ces hypothèses un système composé de deux congruences. Pouvez-vous obtenir la valeur de la clef ?*

1.5 Systèmes affines (en dimension n)

On peut généraliser le système affine en dimension 1 à une dimension n . Ceci permet d'élargir l'espace des clefs et complique les attaques sur texte chiffré. Cependant tous ces systèmes sont faibles par rapport à une attaque sur texte clair connu.

Système de Vigenère

Le système de Vigenère généralise le système de César en dimension n . On prend :

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbf{Z}_{26})^n ,$$

et on pose :

$$E_{\vec{k}}(\vec{p}) = (\vec{p} + \vec{k}) \bmod 26 , \quad D_{\vec{k}}(\vec{c}) = (\vec{c} - \vec{k}) \bmod 26 .$$

Il s'agit d'un premier exemple de *système poly-alphabétique* : un caractère peut être transformé dans des caractères différents selon sa position dans le texte. Par opposition, dans les systèmes qu'on a considéré jusqu'à maintenant chaque caractère est transformé dans un seul caractère de façon indépendante de sa position. Dans ce cas, on parle de *systèmes mono-alphabétiques*. Dans un système mono-alphabétique, le diagramme de distribution des caractères du chiffrement d'un texte français (par exemple) est identique, à une permutation près, à celui du texte clair.

On considère des attaques sur le texte chiffré de ce système. Le nombre de clefs est maintenant $(26)^n$, ce qui est déjà considérable pour $n = 10$. Ainsi une attaque par énumération semble inefficace. On note que si l'on connaît la période n , alors on peut disposer le texte sur n colonnes. Par exemple, si le texte chiffré est :

XUOLMABDHJKZWJM...

et on sait que $n = 5$, on peut disposer le texte de la façon suivante :

| | | | | |
|-----|-----|-----|-----|-----------|
| X | U | O | L | M |
| A | B | D | H | J |
| K | Z | W | J | $M \dots$ |

On sait que les lettres sur la même colonne dépendent du même décalage et on peut donc appliquer une analyse de fréquence : la lettre la plus fréquente dans une colonne correspond probablement à la lettre la plus fréquente dans le langage naturel. On peut ainsi déterminer le décalage correspondant. Pour déterminer la longueur n de la clef, on pourrait essayer d'énumérer les longueurs possibles $n = 1, 2, 3, \dots$ de façon croissante, mais il est possible de réduire fortement les possibilités. Une première approche attribuée à Kasinski (1860) est la suivante :

1. On cherche dans le texte des répétitions du même mot (le plus long le mieux).
2. On calcule les distances entre les occurrences de chaque mot répété. Si un mot répété vient du chiffrement du même mot à partir de la même position alors les distances doivent être des multiples de la période n .
3. On calcule le *pgcd* des distances de mots répétés. On s'attend à que la période n soit parmi les diviseurs du *pgcd*.
4. On applique l'analyse de fréquence aux diviseurs.

Une méthode alternative pour le calcul de la période utilise un *indice de coïncidence* qu'on va définir. Soit n_i le nombre d'occurrences de la lettre i dans un texte avec n lettres. On a donc $n = \sum_{i=0, \dots, 25} n_i$. Si l'on tire 2 lettres parmi les n , la probabilité que les deux lettres soient i est donc :

$$p_i = \frac{\binom{n_i}{2}}{\binom{n}{2}} \quad i = 0, \dots, 25 .$$

L'*indice de coïncidence* (IC) est obtenu en additionnant ces probabilités pour toutes les lettres :

$$IC = \sum_{i=0, \dots, 25} p_i .$$

On remarque que l'indice est *invariant par substitution* des lettres, et donc en particulier par décalage. Il se trouve que les textes dans un langage naturel donné ont un indice de coïncidence caractéristique. Par exemple, pour l'anglais, le français et l'allemand on s'attend à avoir un *IC* de 0,065, 0,074 et 0,072, respectivement. La méthode pour retrouver la longueur de la clef est donc la suivante : on dispose le texte sur un nombre croissant de colonnes jusqu'à arriver à une situation où toutes les colonnes ont un indice de coïncidence proche de celui du langage naturel du texte clair.

Système par transposition (ou permutation)

Le système par transposition (on dit aussi par permutation) comme le système par substitution utilise une permutation comme clef. La différence est que dans le système par substitution on a une permutation sur \mathbf{Z}_{26} qu'on applique lettre par lettre (on a un système mono-alphabétique) alors que dans le système par transposition on a une permutation sur $\{1, \dots, n\}$ qu'on applique au texte chiffré par blocs de n lettres (on a un système poly-alphabétique). On a donc :

$$\mathcal{P} = \mathcal{C} = \Sigma^n, \quad \mathcal{K} = \{\pi :: \{1, \dots, n\} \rightarrow \{1, \dots, n\} \mid \pi \text{ permutation.}\}$$

Si π^{-1} est la permutation inverse de π on a :

$$E_\pi(x_1, \dots, x_n) = (x_{\pi(1)}, \dots, x_{\pi(n)}), \quad D_\pi(x_1, \dots, x_n) = (x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(n)}).$$

De façon plus algébrique, on peut considérer que les fonctions de chiffrement correspondent à des matrices $n \times n$ qui sont obtenues de la matrice identité en permutant les lignes. Par exemple, la permutation :

$$\pi(1) = 2 \quad \pi(2) = 3 \quad \pi(3) = 1,$$

correspond à la matrice :

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Dans un chiffrement par transposition la fréquence des lettres dans le texte chiffré est la même que dans le texte clair. Ainsi il est facile de voir si le chiffrement d'un texte en langue naturelle a été obtenu avec un chiffrement par transposition. Une attaque sur texte chiffré peut utiliser un dictionnaire pour éliminer rapidement les permutations qui génèrent des suites de lettres impossibles. Par ailleurs, on peut sélectionner certains digrammes ou trigrammes qu'on s'attend à trouver dans le texte clair et ensuite chercher des permutations qui maximisent leur fréquence dans le texte chiffré. Enfin, si un texte clair est connu, on peut chercher son anagramme dans le texte chiffré.

Système de Hill

Dans ce système la fonction de chiffrement est déterminée par une *fonction linéaire inversible* sur \mathbf{Z}_{26}^n ou de façon équivalente par une matrice $A \in \mathbf{Z}_{26}[n, n]$ avec un déterminant inversible. Dans ce cas, le chiffrement est simplement la multiplication (modulo 26) de la matrice par un vecteur de n lettres :

$$E_A(\vec{x}) = (A\vec{x}) \bmod 26.$$

Attaque par texte clair connu

Le système de Vigenère, le système par transposition et le système de Hill sont des exemples de systèmes affines en dimension n sur \mathbf{Z}_m . La clef d'un tel système est une matrice inversible $A \in \mathbf{Z}_m[n, n]$ et un vecteur $b \in \mathbf{Z}_m^n$. Le chiffrement est calculé par :

$$E_{(A,b)}(\vec{x}) = (A\vec{x} + b) \bmod m.$$

Les systèmes affines sont sujets à une attaque sur texte clair connu. En effet, si l'attaquant connaît $n + 1$ textes clairs w_i et les textes chiffrés correspondants $c_i = (Aw_i + b) \bmod m$, $i = 0, \dots, n$ alors il peut appliquer la méthode suivante.

1. Soit W une matrice $n \times n$ dont les colonnes sont les différences $(w_i - w_0) \bmod m$, pour $i = 1, \dots, n$.
2. Soit C une matrice $n \times n$ dont les colonnes sont les différences $(c_i - c_0) \bmod m$, pour $i = 1, \dots, n$.
3. On remarque que $c_i - c_0 \equiv A(w_i - w_0) \bmod m$. Ainsi on a $(AW \equiv C) \bmod m$.
4. Si, avec un peu de chance, $\det(W)$ est premier avec m , alors soit w' son inverse et soit $W^{-1} = w' \text{adj}(W)$ (voir appendice B). Alors :

$$(A \equiv CW^{-1}) \bmod m \quad \text{et} \quad (b \equiv (c_0 - Aw_0)) \bmod m.$$

Exercice 4 On suppose savoir qu'un système de Hill sur $(\mathbf{Z}_{26})^2$ chiffre *HAND* avec *FOOT*. Est-ce suffisant pour déterminer la clef ?

1.6 Un exemple de protocole cryptographique

On a déjà mentionné que les systèmes cryptographiques ont plusieurs applications qui vont bien au delà de la simple tâche de protéger la confidentialité d'un message. En général, à partir d'un certain nombre de fonctionnalités cryptographiques de base on peut construire des protocoles qui assurent des fonctionnalités plus compliquées.

On présente (de façon informelle) un exemple d'une telle situation. L'objectif est de concevoir un protocole pour jouer à pile ou face au téléphone en se basant sur la notion de *fonction à sens unique*³ et *résistante aux collisions*.

De façon informelle, on dispose d'une fonction $f : D \rightarrow D$ sur un (grand) domaine fini D telle que :

1. Étant donné x , il est *facile* de calculer $f(x)$.
2. Étant donné y , il est *difficile* de trouver x tel que $f(x) = y$ (inversion difficile).
3. Il est *difficile* de trouver $x_1 \neq x_2$ tels que $f(x_1) = f(x_2)$ (collision difficile).

Pour l'instant, on suppose que facile veut dire en temps polynomial et difficile veut dire qu'on ne connaît pas un algorithme polynomial (ce qui est plus faible que de dire qu'un algorithme polynomial n'existe pas!). Par ailleurs, on fait les hypothèses suivantes :

- A et B veulent jouer à pile ou face au téléphone.
- La ligne de téléphone est 'sécurisée' : A et B sont sûrs de parler entre eux.
- A et B se mettent d'accord pour utiliser une fonction f avec les propriétés décrites ci-dessus.
- A et B ne se font pas confiance (ils viennent de divorcer et doivent décider qui garde la maison...).

Pour décrire le protocole, il convient de fixer un peu de notation :

3. Un résultat central de la cryptographie qui sera évoqué dans les chapitres suivants est que l'existence d'une fonction à sens unique est une condition nécessaire et suffisante à l'existence de systèmes cryptographiques 'pratiques et sûrs'.

| A | B |
|---|--|
| (1) $[x_1 \leftarrow \mathbf{Z}_m : B!f(x_1)]$ | $\rightarrow A?y_1.$ |
| (2) $B?x_2.$ | $\leftarrow [y_2 \leftarrow \{0, 1\} : A!y_2]$ |
| (3) $B!x_1.$ | $\rightarrow A?y_3.$ |
| (4) if $(x_1 \bmod 2 = x_2)$ then LOSE else WIN | if $(f(y_3) = y_1)$ then if $(y_3 \bmod 2 = y_2)$ then WIN else LOSE |

TABLE 1.2 – Un protocole pour jouer à pile ou face au téléphone

- $x \leftarrow D$: on affecte à x une valeur aléatoire du domaine D . Par exemple, si $D = \mathbf{Z}_m = \{0, \dots, m-1\}$ alors $x \leftarrow \mathbf{Z}_m$ affecte à x une valeur dans \mathbf{Z}_m choisie avec probabilité uniforme.
- $A!exp$: on envoie à A un message qui résulte de l'évaluation de l'expression exp .
- $B?x$: on reçoit de B un message dont la valeur est affectée à la variable x .

Avec cette notation, la table 1.2 présente un protocole pour jouer à pile ou face au téléphone.

On ne présente pas une analyse formelle de ce protocole. Mais les questions suivantes expliquent en partie sa conception.

1. On suppose que B a la capacité d'inverser f . Comment B peut-il tricher (en effectuant un calcul additionnel) ?
2. On suppose que A a la capacité de trouver des collisions dans f . Comment A peut-il tricher (en effectuant un calcul additionnel) ?
3. On suppose que le générateur aléatoire de A est biaisé alors que celui de B est équitable. Par exemple, A choisit plus souvent un nombre pair. Les chances que A gagne sont-elles affectées ? Quid si le générateur de B est biaisé et celui de A est équitable ?
4. Maintenant on suppose que les générateurs de A et de B sont biaisés. Les chances que A gagne sont-elles affectées ?

1.7 Sommaire

On a introduit de façon informelle les notions de système cryptographique (pour la confidentialité) et de fonction à sens unique résistante aux collisions. On a décrit un certain nombre de systèmes cryptographiques historiques basés sur les notions de permutation et de fonction affine ainsi qu'un exemple de protocole cryptographique qui repose sur la notion de fonction à sens unique. Tous les systèmes historiques présentés sont sujets à des attaques.

Chapitre 2

Confidentialité parfaite

Tous les systèmes historiques discutés dans le chapitre 1 sont sujets à des attaques. Est-ce possible de construire un système cryptographique pour lequel on peut *prouver* qu’aucune attaque est possible ? Pour répondre à cette question, il faudra d’abord définir exactement ce qui compte comme une attaque.

2.1 Système à masque jetable

Par simplicité on travaille sur un alphabet binaire $\Sigma = \{0, 1\}$. Si le texte clair est rédigé en langage naturel on pourra toujours coder les lettres de l’alphabet en binaire en utilisant, par exemple, un des codages standard (ASCII, ISO, ...).

On considère un système cryptographique dans lequel la clef est au moins aussi longue que le message à chiffrer. Ainsi si le texte clair a longueur au plus n on prend :

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \{0, 1\}^n .$$

Pour chiffrer et déchiffrer on utilise la fonction logique binaire \oplus (*xor* ou *ou exclusif*) :

$$\oplus : \{0, 1\}^2 \rightarrow \{0, 1\} ,$$

dont on rappelle la table de vérité :

| x_1 | x_2 | \oplus |
|-------|-------|----------|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |
| 1 | 0 | 1 |

On étend la fonction *xor* à des vecteurs de n bits :

$$\oplus : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n ,$$

en appliquant le *xor* coordonnée par coordonnée. Ensuite, le système à masque jetable (*one-time pad* en anglais) est défini par :

$$\begin{aligned} E_k : \{0, 1\}^n &\rightarrow \{0, 1\}^n , & E_k(p) &= p \oplus k , \\ D_k : \{0, 1\}^n &\rightarrow \{0, 1\}^n , & D_k(c) &= c \oplus k . \end{aligned}$$

On rappelle que le xor est associatif et commutatif et que, pour tout $x \in \{0, 1\}$, $x \oplus x = 0$ et $x \oplus 0 = x$. Ainsi :

$$D_k(E_k(p)) = (p \oplus k) \oplus k = p .$$

On va montrer que ce système assure une forme de confidentialité maximale qu'on appelle confidentialité parfaite.

2.2 Confidentialité parfaite

Soit $(\mathcal{P}, \mathcal{C}, \mathcal{K}, G, E, D)$ un système cryptographique (définition 1). On va construire un *espace de probabilité* discret (Ω, \mathcal{A}, P) sur ce système et l'utiliser pour définir une notion de confidentialité parfaite.

On prend comme ensemble des expériences Ω le produit cartésien $\mathcal{P} \times \mathcal{K}$ et on suppose que tout sous-ensemble de Ω est un événement. Ainsi l'ensemble des événements est : $\mathcal{A} = \mathcal{P}(\Omega)$. On va utiliser les abréviations suivantes :

- $(p, k) \equiv \{(p, k)\}$ est l'événement où le texte clair p est chiffré avec la clef k .
- $p \equiv \{(p, k) \mid k \in \mathcal{K}\}$ est l'événement où le texte clair p est chiffré.
- $k \equiv \{(p, k) \mid p \in \mathcal{P}\}$ est l'événement où la clef k est utilisée pour chiffrer un texte clair.
- $c \equiv \{(p, k) \mid E_k(p) = c\}$ est l'événement où le texte chiffré est c .

On fait deux hypothèses :

1. Les événements p et k sont *indépendants*, c'est-à-dire

$$P(p \cap k) = P(p, k) = P(p) \cdot P(k) .$$

Cette hypothèse est très naturelle car en général le choix de la clef ne dépend du texte clair.

2. L'attaquant connaît $P(p)$, pour tout texte clair $p \in \mathcal{P}$. Cette hypothèse est en faveur de l'attaquant qui a une connaissance parfaite de la fréquence de tout texte clair (de longueur n).

La définition de confidentialité parfaite repose sur la notion de probabilité conditionnelle. On rappelle que la probabilité de l'événement A conditionnée sur l'événement B (avec $P(B) > 0$) s'écrit $P(A \mid B)$ et qu'on a :

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} .$$

Définition 5 (confidentialité parfaite) *Un système cryptographique assure la confidentialité parfaite si le fait d'observer un texte chiffré c ne fournit aucune information (probabiliste) sur le texte clair correspondant p . En d'autres termes, si pour tout $p \in \mathcal{P}$ et $c \in \mathcal{C}$ avec $P(c) > 0$ on a : $P(p \mid c) = P(p)$.*

Remarque 6 *Le fait que $P(p \mid c) = P(p)$ implique que p et c sont indépendants car :*

$$P(p \cap c)/P(c) = P(p) \quad \text{implique} \quad P(p \cap c) = P(p) \cdot P(c) .$$

Exemple 7 *Soient $\mathcal{P} = \{0, 1\}$, $\mathcal{K} = \{A, B\}$, $\mathcal{C} = \{a, b\}$. La fonction de chiffrement est définie par :*

| p | k | $E_k(p)$ |
|-----|-----|----------|
| 0 | A | a |
| 1 | A | b |
| 0 | B | b |
| 1 | B | a |

La probabilité des textes clairs p , des clefs k et la probabilité dérivée du texte chiffré c :

| p | $P(p)$ | k | $P(k)$ | c | $P(c)$ |
|-----|--------|-----|--------|-----|--------|
| 0 | 1/4 | A | 1/4 | a | 10/16 |
| 1 | 3/4 | B | 3/4 | b | 6/16 |

Les autres probabilités dérivées :

| p | c | $P(p \cap c)$ | $P(p \cap c)/P(c)$ |
|-----|-----|---------------|--------------------|
| 0 | a | 1/16 | 1/10 |
| 1 | a | 9/16 | 9/10 |
| 0 | b | 3/16 | 1/2 |
| 1 | b | 3/16 | 1/2 |

Ce système n'a assure pas la confidentialité parfaite. Par exemple, on suppose que l'attaquant voit le texte chiffré a . La probabilité que le texte clair est 1 est 9/10. Ce qui est différent de 3/4 qui est la probabilité qu'un texte clair est 1.

2.3 Caractérisation de la confidentialité parfaite

La proposition suivante due à Shannon [Sha49] fournit des conditions nécessaires et suffisantes pour avoir la confidentialité parfaite.

Proposition 8 Soit $(\mathcal{P}, \mathcal{C}, \mathcal{K}, G, E, D)$ un système cryptographique tel que $\#\mathcal{P} = \#\mathcal{C} = \#\mathcal{K}$ et pour tout $p \in \mathcal{P}$ et $c \in \mathcal{C}$, $P(p), P(c) > 0$. Alors le système assure la confidentialité parfaite si et seulement si :

1. Les clefs sont sélectionnées avec une probabilité uniforme : $P(k) = 1/\#\mathcal{K}$.
2. $\forall p, c \exists ! k (E_k(p) = c)$.

PREUVE. (\Rightarrow) Pour tout texte clair p , on doit avoir :

$$\#\{E_k(p) \mid k \in \mathcal{K}\} = \#\mathcal{C} .$$

S'il y avait un texte chiffré c qui diffère de $E_k(p)$ pour tout k , alors

$$P(c \mid p) = 0 \neq P(c) > 0 ,$$

ce qui contredit l'hypothèse de confidentialité parfaite. Pourquoi ? Ainsi pour tout texte clair p et texte chiffré c il existe unique $k(p, c) \in \mathcal{K}$ tel que $E_{k(p, c)}(p) = c$. On remarque que $\mathcal{K} = \{k(p, c) \mid p \in \mathcal{P}\}$. Maintenant, on démontre que $P(k(p, c))$ est *constante* par rapport à p , ce qui implique que $P(k(p, c)) = 1/\#\mathcal{K}$. Pour démontrer cette propriété, on *calcule* :

$$\begin{aligned} P(p) &= P(p \mid c) && \text{(par confidentialité parfaite)} \\ &= \frac{P(c \mid p) \cdot P(p)}{P(c)} && \text{(par Bayes)} \\ &= \frac{P(k(p, c)) \cdot P(p)}{P(c)} && \text{(par définition).} \end{aligned}$$

Ainsi $P(k(p, c)) = P(c)$ pour tout texte clair p .

(\Leftarrow) Il suffit d'observer que :

$$\begin{aligned}
 P(p \mid c) &= \frac{P(p) \cdot P(c \mid p)}{P(c)} && \text{(par Bayes)} \\
 &= \frac{P(p) \cdot P(k(p, c))}{\sum_{q \in \mathcal{P}} P(q) \cdot P(k(q, c))} && \text{(par hyp. (2))} \\
 &= \frac{P(p) \cdot (\#\mathcal{K})^{-1}}{(\#\mathcal{K})^{-1} \cdot (\sum_{q \in \mathcal{P}} P(q))} && \text{(par hyp. (1))} \\
 &= P(p) .
 \end{aligned}$$

□

Corollaire 9 *Le système à masque jetable avec choix uniforme des clefs assure la confidentialité parfaite.*

PREUVE. On vérifie sur $\{0, 1\}$ que $\forall c, p \exists! k (p \oplus k = c)$. Ceci se généralise de suite à $\{0, 1\}^n$.

□

Exercice 10 *Vérifier que le système de l'exemple 7 assure la confidentialité parfaite à condition de choisir les clefs avec une probabilité uniforme.*

Remarque 11 *Le système à masque jetable demande une clef aussi longue que le message ce qui n'est pas du tout pratique. Malheureusement cette condition est nécessaire pour obtenir une confidentialité parfaite. On suppose qu'on dispose d'un système cryptographique tel que :*

- $\#\mathcal{K} < \#\mathcal{P}$.
- Pour tout $p \in \mathcal{P}$, $P(p) > 0$.

On a donc moins de clés que de textes à chiffrer. Or un tel système ne peut pas assurer la confidentialité parfaite. En effet soit $c = E_k(p)$ avec $P(k) > 0$. Il doit exister $p' \in \mathcal{P}$ tel que pour tout $k' \in \mathcal{K}$, $p' \neq D_{k'}(c)$. Il suit que $0 = P(p' \mid c) \neq P(p') > 0$.

Remarque 12 *Le système à masque jetable assure la confidentialité parfaite mais il ne garantit pas l'intégrité du message. Par exemple, un attaquant peut compléter les bits du message chiffré. Dans ce cas, le récepteur, après déchiffrement du message, va accepter un message qui est le complémentaire du message originale.*

2.4 Sommaire

On a introduit une notion de confidentialité parfaite qui repose sur la notion de probabilité conditionnelle : le chiffrement est parfait si l'observation du texte chiffré nous apprend rien sur la distribution du texte clair. On obtient ce type de confidentialité avec des systèmes dans lesquels la distribution des textes chiffrés est uniforme. Le prix à payer est qu'il faut avoir autant de clefs que de textes à chiffrer.

Chapitre 3

Distance statistique et équivalence calculatoire

Le système à masque jetable assure la confidentialité parfaite en faisant en sorte que la distribution des chiffréments d'un texte clair soit toujours uniforme. Ainsi les distributions des chiffréments de deux textes clairs de même longueur sont *identiques*. Dans les chapitres suivants, on s'intéresse à une notion de confidentialité plus faible dans l'espoir de pouvoir construire des systèmes cryptographiques *sûrs et pratiques* qui ont cette propriété. Dans ce chapitre, on se pose la question de comment comparer deux distributions. On commence par définir une *distance* entre distributions qu'on appelle *distance statistique*. Cette notion sera ensuite combinée avec deux autres notions : celle d'*algorithme de décision calculable en temps polynomial* (PPT) et celle de fonction *négligeable* pour arriver enfin à une notion d'*équivalence calculatoire* de (familles de) variables aléatoires discrètes. Il s'agit d'un petit tour de force qui, dans l'état actuel de nos connaissances, semble inévitable pour arriver à donner une définition robuste et rigoureuse des propriétés de sécurité (dont la confidentialité et l'intégrité).

3.1 Distance statistique

On dénote par X, Y, \dots des variables aléatoires discrètes (v.a.d) dont l'image est contenue dans un ensemble fini R .

Définition 13 (distance statistique) Soient $X, Y : \Omega \rightarrow R$ v.a.d. avec R fini. On définit leur distance statistique par :

$$\Delta(X, Y) = \frac{1}{2} \sum_{x \in R} |P(X = x) - P(Y = x)| .$$

Si $X : \Omega \rightarrow R$ est une v.a.d., soit $f_X : R \rightarrow [0, 1]$ la distribution associée qui est définie par $f_X(x) = P(X = x)$. La fonction Δ ne dépend que des distributions associées aux v.a.d. et sur les distributions elle est une distance.¹

Proposition 14 Soient $X, Y, Z : \Omega \rightarrow R$ v.a.d. avec R fini. Alors :

1. $\Delta(X, Y) \in [0, 1]$.

1. Il s'agit d'un cas particulier de la distance enduite par la norme-1 sur les fonctions.

2. $\Delta(X, Y) = 0$ ssi $f_X = f_Y$.
3. $\Delta(X, Y) = \Delta(Y, X)$.
4. $\Delta(X, Z) \leq \Delta(X, Y) + \Delta(Y, Z)$.

La preuve est un exercice simple. La caractérisation suivante de la distance Δ utilise le fait que R est fini.

Proposition 15 Soient $X, Y : \Omega \rightarrow R$ v.a.d. avec R fini. Alors :

$$\Delta(X, Y) = \max_{R' \subseteq R} |P(X \in R') - P(Y \in R')| .$$

PREUVE. Soient :

$$\begin{aligned} R_X &= \{x \in R \mid P(X = x) > P(Y = x)\} , \\ R_Y &= \{x \in R \mid P(X = x) < P(Y = x)\} , \\ R_Z &= \{x \in R \mid P(X = x) = P(Y = x)\} . \end{aligned}$$

On remarque que :

$$\begin{aligned} 1 &= \sum_{x \in R} P(X = x) \\ &= \sum_{x \in R_X} P(Y = x) + (P(X = x) - P(Y = x)) + \sum_{x \in R_Y} P(X = x) + \sum_{x \in R_Z} P(X = x) \\ &= \sum_{x \in R_X} P(Y = x) + \sum_{x \in R_Y} P(X = x) + (P(Y = x) - P(X = x)) + \sum_{x \in R_Z} P(Y = x) \\ &= \sum_{x \in R} P(Y = x) = 1 . \end{aligned}$$

On pose :

$$\begin{aligned} \delta_X &= \sum_{x \in R_X} |P(X = x) - P(Y = x)| , \\ \delta_Y &= \sum_{x \in R_Y} |P(Y = x) - P(X = x)| , \end{aligned}$$

et on dérive que $\delta_X = \delta_Y$ et

$$\Delta(X, Y) = \frac{1}{2}(\delta_X + \delta_Y) = \delta_X = \delta_Y .$$

On remarque aussi que :

$$\delta_X = |P(X \in R_X) - P(Y \in R_Y)| = |P(X \in R_Y) - P(Y \in R_Y)| = \delta_Y .$$

Donc en prenant $R' = R_X$ (ou de façon équivalente $R' = R_Y$) on obtient :

$$\Delta(X, Y) = |P(X \in R') - P(Y \in R')| .$$

Soit maintenant $R' \subseteq R$. On partitionne R' en 3 ensembles : $R'_X = R' \cap R_X$, $R'_Y = R' \cap R_Y$ et $R'_Z = R' \cap R_Z$ et on montre que :

$$\begin{aligned} &|P(X \in R') - P(Y \in R')| \\ &\leq \max(|P(X \in R'_X) - P(Y \in R'_X)|, |P(X \in R'_Y) - P(Y \in R'_Y)|) \\ &\leq \max(\delta_X, \delta_Y) \\ &= \Delta(X, Y) . \end{aligned}$$

□

Considérons maintenant la situation où l'on compose les v.a.d. X, Y avec une fonction $f : R \rightarrow R'$.

Proposition 16 Soient $X, Y : \Omega \rightarrow R$ v.a.d. avec R fini et $f : R \rightarrow R'$ une fonction (avec R' fini). Alors on abrège $\Delta(f \circ X, f \circ Y)$ en $\Delta_f(X, Y)$ et on a :

$$\Delta_f(X, Y) \leq \Delta(X, Y) .$$

PREUVE. Par la proposition 15, on sait que :

$$\Delta(f \circ X, f \circ Y) = \max_{R' \subseteq R} |P(f \circ X \in R') - P(f \circ Y \in R')| .$$

Si le max est atteint pour R' , on pose $R'' = f^{-1}(R')$ et on obtient :

$$\Delta(f \circ X, f \circ Y) = |P(X \in R'') - P(Y \in R'')| \leq \Delta(X, Y) .$$

□

Supposons maintenant que $R' = \{0, 1\}$. Donc la fonction f prend un échantillon d'une v.a.d. et produit une valeur 0 ou 1.

Proposition 17 Soient $X, Y : \Omega \rightarrow R$ v.a.d. avec R fini et $f : R \rightarrow \{0, 1\}$ une fonction. Alors :

$$\Delta_f(X, Y) = |P(f \circ X = 1) - P(f \circ Y = 1)| = |P(f \circ X = 0) - P(f \circ Y = 0)| .$$

PREUVE. Soient $p = P(f \circ X = 1)$ et $q = P(f \circ Y = 1)$. Alors :

$$\begin{aligned} \Delta_f(X, Y) &= \frac{1}{2}(|p - q| + |(1 - p) - (1 - q)|) \\ &= |p - q| \\ &= |P(f \circ X = 1) - P(f \circ Y = 1)| . \end{aligned}$$

□

Proposition 18 Soient $X, Y : \Omega \rightarrow R$ v.a.d. avec R fini. Alors il existe $f : R \rightarrow \{0, 1\}$ telle que :

$$\Delta(X, Y) = \max_{f: R \rightarrow \{0, 1\}} \Delta_f(X, Y) .$$

PREUVE. Par la proposition 15, on sait que :

$$\Delta(X, Y) = \max_{R' \subseteq R} |P(X \in R') - P(Y \in R')| .$$

Il suffit donc de prendre $f_{R'}$ comme la fonction caractéristique de l'ensemble R' qui maximise la différence :

$$\Delta(X, Y) = |P(X \in R') - P(Y \in R')| = |P(f_{R'} \circ X = 1) - P(f_{R'} \circ Y = 1)| .$$

□

Exemple 19 Soit X une v.a.d. qui échantillonne n -bits avec probabilité uniforme et soit $Y = X^c$. Soit π_1 la fonction qui prend une suite de n -bits et retourne le premier bit (la première projection). On a :

$$\Delta_{\pi_1}(X, Y) = |P(\pi_1 \circ X = 1) - P(\pi_1 \circ Y = 1)| = 0 .$$

Donc la distance statistique des v.a.d. $\pi_1 \circ X$ et $\pi_1 \circ Y$ est 0 alors que les v.a.d. $\pi_1 \circ X$ et $\pi_1 \circ Y$ diffèrent avec probabilité 1, à savoir : $P(\pi_1 \circ X \neq \pi_1 \circ Y) = 1$. Plus en général, on a que la distance statistique des v.a.d. X et Y est 0 :

$$\Delta(X, Y) = \frac{1}{2} \sum_{x \in 2^n} |P(X = x) - P(Y = x)| = 0 .$$

Cependant, la situation change si, par exemple, on suppose que X échantillonne n -bits avec une probabilité $3/4$ que le premier bit soit 1. Dans ce cas on a :

$$|P(\pi_1 \circ X = 1) - P(\pi_1 \circ Y = 1)| = 1/2 .$$

Exemple 20 Certaines propriétés de sécurité sont formalisées comme un jeu. On discute un exemple dans le contexte traité dans ce chapitre. Soient $X_0, X_1 : \Omega \rightarrow R$ v.a.d. On évalue l'efficacité (on parle aussi d'avantage) d'un test $f : R \rightarrow \{0, 1\}$ qui cherche à distinguer un échantillon de X_0 d'un échantillon de X_1 . Pour ce faire, on imagine que f reçoit avec probabilité $1/2$ un échantillon de X_0 et avec probabilité $1/2$ un échantillon de X_1 et que le test 'gagne' s'il devine l'origine de l'échantillon. Remarquons qu'un test qui répond 1 avec probabilité p et 0 avec probabilité $(1 - p)$ a une probabilité $1/2$ de gagner. Ainsi si q est la probabilité que le test gagne on s'intéresse à la quantité $|q - 1/2|$: plus cette quantité est grande plus le test est efficace. Pour formaliser le jeu, on introduit une nouvelle v.a.d. Z . L'idée est que chaque échantillon a maintenant un marqueur 0 ou 1 qui permet de déterminer son origine. Ce marqueur est invisible au test mais il nous permet de savoir si le test a déterminé correctement l'origine de l'échantillon. Ainsi on définit $\Omega' = \Omega \times \{0, 1\}$, $R' = R \times \{0, 1\}$ $Z : \Omega' \rightarrow R'$ v.a.d. telle que $Z(\omega, i) = (X_i(\omega), i)$ et $P(Z = (x, i)) = \frac{1}{2}P(X_i = x)$. Ensuite, la fonction Test est définie par :

$$\begin{aligned} f \times id : R' &\rightarrow (\{0, 1\} \times \{0, 1\}) & (f \times id)(x, i) &= (f(x), i) \\ eq : (\{0, 1\} \times \{0, 1\}) &\rightarrow \{0, 1\} & eq(i, j) &= \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases} \\ \text{Test} &= eq \circ (f \times id) \circ Z . \end{aligned}$$

On va montrer que :

$$|P(\text{Test} = 1) - \frac{1}{2}| = \frac{1}{2} |P(f \circ X_0 = 1) - P(f \circ X_1 = 1)| = \frac{1}{2} \Delta_f(X_0, X_1) . \quad (3.1)$$

Cette équation est remarquable car elle nous dit que l'efficacité du test f est la moitié de la distance statistique des v.a.d. $f \circ X_0$ et $f \circ X_1$. Soient $p_i = P(f \circ X_i = 1)$ pour $i = 0, 1$. On calcule $P(\text{Test} = 1)$ en conditionnant sur i :

$$P(\text{Test} = 1) = \frac{1}{2} \cdot (P(f \circ X_0 = 1) + P(f \circ X_1 = 1)) = \frac{1}{2} \cdot ((1 - p_0) + p_1) .$$

Il suit que :

$$|P(\text{Test} = 1) - \frac{1}{2}| = |\frac{1}{2}((1 - p_0) + p_1) - \frac{1}{2}| = \frac{1}{2} \cdot |p_0 - p_1| .$$

3.2 Temps probabiliste polynomial

Une façon d'affaiblir la notion de confidentialité parfaite est de limiter le pouvoir de l'attaquant. Par exemple, on s'attend à qu'un attaquant ne puisse pas énumérer tous les

chiffrements d'un message donné. Pour définir la notion d'attaquant 'raisonnable', on peut se baser sur l'idée qu'un algorithme efficace est un algorithme qui calcule en temps polynomial. On rappelle brièvement les définitions.

Définition 21 (*O*-notation) Soient $f, g : \mathbf{N} \rightarrow \mathbf{N}$ deux fonctions sur les nombres naturels. On dit que f est $O(g)$ si :

$$\exists k, n_0 \geq 0 \ \forall n \geq n_0 \ f(n) \leq k \cdot g(n) .$$

Définition 22 (fonction de coût) Soit A un algorithme qui termine et $c_A : \mathbf{N} \rightarrow \mathbf{N}$ une fonction telle que $c_A(n)$ est le coût maximal d'une exécution de l'algorithme A sur une entrée de taille au plus n .

Typiquement, la taille d'une entrée est le nombre de bits nécessaires à sa représentation et le coût d'une exécution est le *temps* mesuré comme le nombre d'étapes élémentaires de calcul.²

Définition 23 (algorithme polynomial) Un algorithme A est polynomial (en temps) s'il existe $d \geq 0$ tel que sa fonction de coût c_A est $O(n^d)$.

Définition 24 (algorithme PPT) Un algorithme PPT (probabilistic polynomial time) est un algorithme qui calcule en temps polynomial dans la taille de son entrée et qui peut utiliser pendant le calcul un générateur de bits aléatoires non biaisé.

Si les seuls résultats de l'algorithme PPT sont 0 ou 1 on parle aussi d'*algorithme de décision PPT*. Dans la suite du cours, on fera l'hypothèse qu'un attaquant raisonnable est un algorithme de décision PPT.

Remarque 25 1. On ne sait pas si le fait d'avoir accès à une source aléatoire augmente vraiment le pouvoir de l'attaquant. Dans le doute, on donne cette possibilité à l'attaquant.

2. La notion d'algorithme efficace repose sur l'idée que la complexité asymptotique d'un algorithme est une fonction de la taille de l'entrée. Pour la même raison la notion d'attaquant raisonnable sera typiquement fonction de la taille de la clef du système cryptographique. On va donc analyser non pas un système cryptographique mais une famille de systèmes cryptographiques indexés, par exemple, sur la taille de la clef.

3. Dans certaines définitions qui vont suivre, on considère un algorithme A qui peut utiliser la fonction g comme un oracle. On dénote cet algorithme par A^g . Il n'y a rien de mystérieux dans la notion d'oracle. La fonction g est comme une fonction de bibliothèque dont on connaît la fonctionnalité mais pas le code. L'algorithme prépare une entrée x pour g , appelle g et obtient (en temps constant) la réponse $g(x)$. Par exemple, la fonction g pourrait être une fonction de chiffrement qui dépend d'une clef secrète k . Dans ce cas, un attaquant A^g est un algorithme qui peut chiffrer un nombre polynomial de textes clairs sans pour autant connaître la clef secrète k .

2. La notion d'étape élémentaire dépend du modèle de calcul mais il s'avère que la notion de temps polynomial est assez robuste pour ne pas dépendre de ce choix.

3.3 Fonctions négligeables

En complexité asymptotique, on sait que la composition séquentielle de deux algorithmes ‘efficaces’ (c’est à dire PPT) est encore un algorithme efficace. Cette propriété est basée sur l’observation que la composition de deux polynômes est encore un polynôme. On souhaite suivre une approche similaire pour définir une notion de ‘quantité négligeable’. En particulier, on s’attend à que la somme de deux quantités négligeables soit encore négligeable, et pour avoir une telle propriété il convient de voir une quantité négligeable comme une fonction.

Définition 26 Une fonction $\epsilon : \mathbf{N} \rightarrow \mathbf{N}$ est négligeable si pour tout $k \geq 0$, ϵ est $O(n^{-k})$. En d’autres termes :

$$\forall k \geq 0 \exists c, n_0 \in \mathbf{N} \forall n \geq n_0 \quad (\epsilon(n) \leq \frac{c}{n^k}) .$$

De façon plus *informelle*, on dit que pour tout polynôme, ϵ décroît plus vite que l’inverse du polynôme. La proposition suivante montre que les fonctions négligeables ont certaines propriétés de stabilité ; ces propriétés permettent ensuite d’avoir une notion d’équivalence de (familles de) v.a.d. qui est robuste.

Proposition 27 1. Si ϵ_1 et ϵ_2 sont négligeables alors $\epsilon_1 + \epsilon_2$ est négligeable.

2. Si ϵ est négligeable et p est un polynôme dans une variable alors $\epsilon'(n) = p(n) \cdot \epsilon(n)$ est encore négligeable.

PREUVE. (1) Soient c_1, n_1, c_2, n_2 les constantes pour ϵ_1 et ϵ_2 . Il suffit de prendre $c = 2 \cdot \max(c_1, c_2)$, $n = \max(n_1, n_2)$.

(2) Soit $p(n) \leq c_1 n^{k_1}$ pour $n \geq n_1$. Prenons $k' = k + k_1$. On a c_2, n_2 tel que $\epsilon(n) \leq c_2 / n^{k+k_1}$. Donc pour tout $n \geq \max(n_1, n_2)$ on a $p(n)\epsilon(n) \leq c_1 c_2 / n^k$. \square

3.4 Équivalence calculatoire

Considérons le *chiffrement d’un message m* . Ce chiffrement dépend du *paramètre de sécurité n* . De plus, si l’on fixe le paramètre de sécurité, le chiffrement va dépendre du *choix aléatoire de la clé*. Pour un n fixé, le chiffrement du message m est donc une *variable aléatoire discrète* (v.a.d.) X_n ; et si n est *variable*, le chiffrement du message m est une *famille de v.a.d.*

$$\{X_n\}_{n \in \mathbf{N}} \quad (\text{famille de v.a.d.})$$

En anglais, on dit *probability ensemble* ou simplement *ensemble*.

Définition 28 On dit que deux familles de v.a.d. $\mathbf{X} = \{X_n\}_{n \in \mathbf{N}}$ et $\mathbf{Y} = \{Y_n\}_{n \in \mathbf{N}}$ sont équivalentes (au sens calculatoire) et on écrit $\mathbf{X} \sim \mathbf{Y}$ si pour tout algorithme de décision PPT A (polynomial en n) la fonction :

$$\epsilon(n) = \Delta_A(X_n, Y_n) = |P(A \circ X_n = 1) - P(A \circ Y_n = 1)|$$

est négligeable.

Remarque 29 1. Ici on suppose implicitement que les v.a.d. X_n, Y_n sont définies sur un même espace de probabilités Ω_n et que leurs images sont contenues dans un ensemble fini R_n dont les éléments ont une taille polynomial en n . On suppose aussi que l'algorithme de décision A reçoit en argument le paramètre de sécurité n et connaît donc le domaine R_n dont il va examiner un échantillon.

2. La notion d'équivalence calculatoire est évidemment réflexive et symétrique. Pour montrer sa transitivité on utilise le fait que la somme de deux fonctions négligeables est négligeable. Donc le choix de considérer des familles de v.a.d. et de mesurer leur proximité par une fonction est essentiel pour pouvoir définir une relation d'équivalence. Par exemple, si on avait dit que deux v.a.d. sont proches si leur distance statistique est inférieure à 2^{-80} on n'aurait pas une relation d'équivalence sur les v.a.d. ($\Delta(X, Y) \leq 2^{-80}$ et $\Delta(Y, Z) \leq 2^{-80}$ n'implique pas $\Delta(X, Z) \leq 2^{-80}$).

La notion d'équivalence calculatoire se base sur l'analyse d'un seul échantillon de la v.a.d. Que se passe-t-il si l'on peut disposer de m échantillons indépendants? Et bien rien ne change tant que :

- le nombre d'échantillons est borné par un polynôme dans le paramètre de sécurité.
- il y a un algorithme PPT pour échantillonner la v.a.d. (on dit que la v.a.d. est échantillonnable).

La preuve de ce fait consiste à construire des familles de v.a.d. *hybrides* qui interpolent les deux familles qui nous intéressent (on parle d'*argument hybride*). On considère une situation *simplifiée* dans laquelle on traite deux v.a.d. (au lieu de deux familles de v.a.d.) et l'on considère un algorithme de décision (au lieu d'un algorithme de décision PPT).

Proposition 30 (argument hybride) Soient X, Y deux v.a.d. échantillonnables telles que pour tout A algorithme de décision :

$$\Delta_A(X, Y) = |P(A \circ X = 1) - P(A \circ Y = 1)| \leq \epsilon .$$

Alors les v.a.d. (X, X) et (Y, Y) (on échantillonne 2 fois de façon indépendante) ont la propriété que pour tout A' algorithme de décision :

$$\Delta_{A'}((X, X), (Y, Y)) \leq 2 \cdot \epsilon .$$

PREUVE. On utilise l'inégalité triangulaire :

$$\begin{aligned} & \Delta_{A'}((X, X), (Y, Y)) \\ &= |P(A' \circ (X, X) = 1) - P(A' \circ (X, Y) = 1) + P(A' \circ (X, Y) = 1) - P(A' \circ (Y, Y) = 1)| \\ &\leq \Delta_{A'}((X, X), (X, Y)) + \Delta_{A'}((X, Y), (Y, Y)) . \end{aligned}$$

Si on pose $A_1 = A' \circ (X, _)$, on a encore un algorithme de décision (hyp. X échantillonnable) et

$$\Delta_{A'}((X, X), (X, Y)) = \Delta_{A_1}(X, Y) \leq \epsilon .$$

De façon similaire, si on pose $A_2 = A' \circ (_, Y)$ alors on dérive :

$$\Delta_{A'}((X, Y), (Y, Y)) = \Delta_{A_2}(X, Y) \leq \epsilon .$$

□

Dans le cas général, on suppose que l'algorithme de décision A' reçoit $p(n)$ échantillons indépendants qui viennent soit de la v.a.d. X_n soit de la v.a.d. Y_n et on montre que la probabilité de les distinguer est bornée par $\epsilon'(n) = p(n) \cdot \epsilon(n)$ où $\epsilon(n)$ est une fonction négligeable. Par la proposition 27(2), on peut donc conclure que la borne $\epsilon'(n)$ est aussi négligeable.

Les *définitions de sécurité* qui vont suivre (1-confidentialité, CPA-confidentialité, intégrité...) sont des *variations* sur la notion d'équivalence; on dispose de deux familles de v.a.d. et on demande à que toute *fonction PPT* ait une probabilité négligeable de les distinguer. Dans ce contexte, les *preuves de sécurité* consistent à montrer que si deux familles de v.a.d. sont équivalentes alors deux autres familles le sont aussi.

3.5 Sommaire

La notion de distance statistique peut être utilisée pour mesurer la différence entre les chiffrements possibles de deux messages. Si la distance est 0, on se retrouve dans une situation de confidentialité parfaite. L'affaiblissement de cette notion passe par un mélange délicat de la notion d'attaquant PPT et de fonction négligeable et mène à la notion d'équivalence calculatoire de familles de v.a.d. Cette notion constitue le fondement des constructions cryptographiques présentées dans les prochains chapitres.

Chapitre 4

Générateurs pseudo-aléatoires de nombres (*PRG*) et 1-confidentialité

On utilise la notion d'équivalence calculatoire (chapitre 3) pour définir la notion de générateur pseudo-aléatoire de nombres (PRG pour *pseudo-random number generator*). Il s'agit d'une fonction qui est capable de produire de façon efficace une 'longue' suite de bits à partir d'une 'petite' suite de bits aléatoires de façon telle que la suite longue est équivalente au sens du chapitre 3 à une suite de bits aléatoires.

On peut utiliser un PRG pour construire une variante 'pratique' du système à masque jetable (chapitre 2) : la clef du système est la petite suite et pour chiffrer on fait le xor du message avec la longue suite générée par le PRG à partir de la clef. Dans un tel système, les distributions des chiffrements de deux messages de même longueur sont équivalentes. Il s'agit d'un affaiblissement de la notion de confidentialité parfaite qu'on appelle 1-confidentialité.

4.1 PRG

Un *générateur pseudo-aléatoire de nombres* (PRG) est une famille de fonctions. La fonction d'indice n prend n bits (aléatoires) et produit $\ell(n) > n$ bits qui semblent aléatoires à un *attaquant PPT*.

Définition 31 (PRG) *Un PRG est une famille de fonctions :*

$$\mathcal{G}_n : 2^n \rightarrow 2^{\ell(n)} \quad n \in \mathbf{N}, \ell(n) > n ,$$

telle que les deux familles de v.a.d suivantes sont équivalentes :

$$X_n = [k \leftarrow 2^n : \mathcal{G}_n(k)] \sim [x \leftarrow 2^{\ell(n)} : x] = Y_n .$$

et les fonctions \mathcal{G}_n sont calculables en temps polynomial déterministe.

Explicitons la différence entre X_n et Y_n . Soit $y \in 2^{\ell(n)}$. Alors :

- $P(X_n = y)$ est la probabilité qu'en tirant k dans 2^n avec probabilité uniforme on obtient $\mathcal{G}_n(k) = y$
- $P(Y_n = y) = \frac{1}{2^{\ell(n)}}.$

On peut remarquer que $\#im(\mathcal{G}) \leq 2^n$. Ainsi une attaque (non polynomiale) pourrait être la suivante : sur un échantillon y , si $y \notin im(\mathcal{G})$ alors on est sûr que l'échantillon vient de Y_n et pas de X_n .

L'état de l'art est qu'on sait construire un PRG à partir de n'importe quel PRG tel que $\ell(n) \geq n + 1$ et aussi à partir d'une fonction à sens unique (intuition dans la section 1.6 et définition 38 à suivre). Par contre, on ne sait pas si ces objets existent (SIC). Néanmoins la définition de PRG est intéressante car elle nous donne une méthode pour montrer qu'une certaine famille de fonctions n'est *pas* un PRG. Il suffit de définir un algorithme PPT qui arrive à distinguer avec une probabilité non-négligeable la fonction en question de la distribution uniforme. En pratique, on commence par utiliser des batteries de tests disponibles dans le domaine public (par exemple, les *tests Diehard*).

Exercice 32 *Considérons la fonction \mathcal{M}_n qui prend en entrée n bits $x_0 \cdots x_{n-1}$ les considère comme la représentation en base 2 d'un nombre $x \in [0, 2^n - 1]$ et produit en sortie $2n$ bits $y_0 \cdots y_{2n-1}$ bits qui correspondent à la représentation en base 2 du nombre x^2 . La famille $\{\mathcal{M}_n\}_{n \in \mathbf{N}}$ est-elle un PRG ?*

4.2 1-Confidentialité

On va réviser la définition 1 de système cryptographique et introduire une notion de 1-confidentialité basée sur la notion d'équivalence calculatoire.

Plutôt qu'un système cryptographique, on considère une *famille* de systèmes cryptographiques indexés sur un paramètre de sécurité n . Pour fixer les idées on peut supposer que ce paramètre n est le nombre de bits nécessaires à la représentation d'une clef. Pour chaque n on a donc un espace de clefs \mathcal{K}_n (typiquement n bits), un espace de textes clairs \mathcal{P}_n et un espace de textes chiffrés \mathcal{C}_n . On suppose que la taille d'un texte clair dans \mathcal{P}_n ou d'un texte chiffré dans \mathcal{C}_n est polynomiale en n . Les algorithmes G , E et D pour générer une clef, chiffrer et déchiffrer sont aussi indexés sur n . On a donc :

$$\begin{aligned} G_n &: 1 \rightarrow \mathcal{K}_n && (\text{générateur}), \\ E_n &: \mathcal{K}_n \times \mathcal{P}_n \rightarrow \mathcal{C}_n && (\text{chiffrement}), \\ D_n &: \mathcal{K}_n \times \mathcal{C}_n \rightarrow \mathcal{P}_n && (\text{déchiffrement}), \end{aligned}$$

tels que, pour tout $n \in \mathbf{N}, k \in \mathcal{K}_n, p \in \mathcal{P}_n : D_n(k, E_n(k, p)) = p$. On suppose que les algorithmes G , E et D sont PPT.

On souhaite maintenant définir dans quel sens cette famille de systèmes assure la confidentialité. Dans le système à *masque jetable*, les distributions des chiffrements de deux messages différents sont *identiques*. Il en suit que le chiffrement d'un message ne dévoile aucune information sur le message clair. On introduit un affaiblissement de cette notion que l'on appelle *confidentialité pour 1 message* (abrégé en 1-confidentialité). L'idée est que les distributions des chiffrements de deux messages différents (de même taille) sont équivalentes.

Définition 33 (1-confidentialité) *Soient $m_n, m'_n \in \mathcal{P}_n$ avec $|m_n| = |m'_n|$ deux familles de messages de même longueur qui peuvent être générées par un algorithme PPT. On définit deux familles de v.a.d. par :*

$$\begin{aligned} X_n &= [k \leftarrow G_n(1) : (m_n, m'_n, E_n(k, m_n))] \\ Y_n &= [k \leftarrow G_n(1) : (m_n, m'_n, E_n(k, m'_n))] . \end{aligned}$$

La v.a.d X_n diffère de la v.a.d. Y_n par le fait que sa troisième composante est un chiffrement de m_n plutôt que de m'_n . On dit que le système assure la 1-confidentialité si :

$$\{X_n\}_{n \in \mathbf{N}} \sim \{Y_n\}_{n \in \mathbf{N}} .$$

En développant la définition d'équivalence, on dérive que pour tout algorithme PPT A la fonction :

$$\epsilon(n) = |P(A \circ X_n = 1) - P(A \circ Y_n = 1)|$$

est négligeable. Intuitivement, un algorithme efficace n'arrive pas à distinguer les chiffrements de deux messages de même longueur.

Un PRG nous donne une suite de bits qui semble aléatoire. Si on utilise cette suite dans un système à masque jetable, on obtient un système qui assure la 1-confidentialité.

Proposition 34 *Si l'on dispose d'un PRG $\mathcal{G}_n : 2^n \rightarrow 2^{\ell(n)}$ (à ne pas confondre avec les générateur de clé) alors on peut construire un système qui assure la 1-confidentialité de messages de longueur $\ell(n)$ en définissant :*

$$\begin{aligned} E_k(m) &= \mathcal{G}_n(k) \oplus m & k \in 2^n, m \in 2^{\ell(n)} \\ D_k(c) &= \mathcal{G}_n(k) \oplus c & k \in 2^n, c \in 2^{\ell(n)} . \end{aligned}$$

PREUVE. Soient m_1, m_2 deux messages de longueur $\ell(n)$. On sait que (masque jetable) :

$$U_{\ell(n)} \oplus m_1 = U_{\ell(n)} \oplus m_2 .$$

Par ailleurs, on doit avoir pour $i = 1, 2$:

$$(\mathcal{G} \circ U_n) \oplus m_i \sim U_{\ell(n)} \oplus m_i$$

autrement, \mathcal{G} n'est pas un PRG. □

Remarque 35 *Dans une utilisation pratique du système, les deux agents qui partagent la clef $k \in 2^n$ peuvent échanger plusieurs messages m_1, \dots, m_k à condition que la taille total de ces messages n'excède pas $\ell(n)$. Pour ce faire, chaque agent doit garder un compteur pour savoir quelle portion de la suite $\mathcal{G}_n(k)$ utiliser. Comme pour le système à masque jetable, il faut veiller à ne jamais utiliser la même portion du masque pour deux messages différents.*

Remarque 36 *L'affirmation que la probabilité d'attaque est négligeable doit être prise avec un grain de sel...*

- La probabilité d'attaque est négligeable si un PRG existe.
- La probabilité d'attaque est négligeable si l'attaquant réel est conforme au modèle 'mathématique'.
- En pratique, on peut convenir du fait que la probabilité d'attaque doit être inférieure à 2^{-80} . Il faut alors s'assurer que le paramètre de sécurité qui garantit un tel niveau de sécurité a une taille raisonnable.

Remarque 37 *Le '1' dans la définition de 1-confidentialité est là pour nous rappeler qu'on fait l'hypothèse (restrictive) que l'attaquant manipule 1 seul chiffrement. Si, par exemple, l'attaquant connaît le chiffrement c d'un message m et si la fonction de chiffrement est déterministe alors l'attaquant n'aura pas de mal à distinguer le chiffrement de m du chiffrement d'un message m' différent de m .*

4.3 PRG : théorie et pratique

En *théorie*, on peut construire un PRG à partir d'une *fonction à sens unique* (voir [KL07]).

Définition 38 (fonction à sens unique) On dénote par 2^* les mots finis sur l'alphabet $\{0, 1\}$. Une fonction $f : 2^* \rightarrow 2^*$ est à sens unique si elle est :

1. facile à calculer, à savoir calculable en PPT,
2. difficile à inverser, pour tout $g : 2^* \rightarrow 2^*$ PPT, la fonction

$$\epsilon(n) = P[x \leftarrow 2^n : f(g(f(x))) = f(x)]$$

est négligeable.

De façon plus informelle, on dit que la probabilité de trouver une image inverse est négligeable.

La construction d'un PRG à partir d'une fonction à sens unique n'est pas très efficace. Par ailleurs, l'existence d'une fonction à sens unique est un problème ouvert.

Une preuve d'existence d'une fonction à sens unique aurait un impact majeur dans la théorie de la complexité du calcul. On dénote par **P** la classe des problèmes de décision qui admettent un algorithme polynomial déterministe (voir définition 23). Par ailleurs, on dénote par **NP** la classe des problèmes de décision pour lesquels on sait *vérifier* la correction d'une solution positive en temps polynomial déterministe.

Un exemple typique de problème dans **P** est le problème de l'évaluation d'une formule du calcul propositionnel : on prend une formule et une affectation de valeurs booléennes aux variables de la formule et on décide si l'évaluation de la formule par rapport à l'affectation est 0 ou 1.

Un exemple typique de problème dans **NP** est le problème de la *satisfiabilité d'une formule* : on prend en entrée une formule du calcul propositionnel et on décide si elle est satisfiable. On ne connaît pas d'algorithme polynomial pour décider ce problème, mais on peut vérifier une solution positive du problème, c'est-à-dire une affectation qui satisfait la formule, en temps polynomial en utilisant l'algorithme pour l'évaluation d'une formule.

Proposition 39 Si une fonction à sens unique existe alors $\mathbf{P} \subset \mathbf{NP}$ (inclusion stricte).

PREUVE. On donne une idée de l'argument. Soit f une fonction candidate à être une fonction à sens unique. Sans perte de généralité, on peut supposer qu'il y a une polynôme p tel que si $f(x) = y$ alors $|x| \leq p(|y|)$ où $|x|$ et $|y|$ sont les tailles de x et y , respectivement. On montre que si $\mathbf{P} = \mathbf{NP}$ alors on a un algorithme efficace pour calculer l'image inverse de f . On remarque que le langage suivant est dans **NP** :

$$L_f = \{(z, y) \mid \exists x \text{ (} z \text{ préfixe de } x \text{ et } f(x) = y)\}.$$

Pour *vérifier* que $(z, y) \in L_f$ il suffit de fournir un x (dont la taille est polynomiale en celle de y) tel que z est un préfixe de x et $f(x) = y$ (la valeur x joue ici le rôle de 'temoin', exactement comme une affectation est un temoin de la satisfiabilité d'une formule). Si maintenant $\mathbf{P} = \mathbf{NP}$ et $f(x) = y$ on a un algorithme efficace pour trouver une image inverse de y : (1) on sait que $(\epsilon, y) \in L_f$ et (2) si $(z, y) \in L_f$ et $f(z) \neq y$ alors on cherche $i \in \{0, 1\}$ tel que $(zi, y) \in L_f$ et on itère (2). \square

Remarque 40 La condition $P \subset NP$ est seulement nécessaire à l'existence d'une fonction à sens unique. On ne sait pas construire une fonction à sens unique en supposant $P \subset NP$.

On sait construire une fonction à sens unique à partir de :

- Un PRG.
- Un système cryptographique qui assure la 1-confidentialité.
- Un système qui assure l'authentification (MAC) (chapitre 6, à venir).

Donc l'existence d'une fonction à sens unique est une condition *nécessaire et suffisante* pour garantir les propriétés de confidentialité et intégrité de la 'cryptographie moderne'. . . A défaut d'avoir des preuves on a des *fonctions candidates* au titre de fonction à sens unique dont voici une petite liste.

- Le *produit* de deux nombres premiers de taille comparable. Pour inverser, il faut *factoriser* (définition A.3).
- L'*exposant modulaire*. Pour inverser, il faut calculer le *logarithme discret*.
- Le problème de la *somme de sous-ensembles* : pour $x_1, \dots, x_n \in \mathbf{Z}$ et $J \subseteq \{1, \dots, n\}$ on considère la fonction :

$$(x_1, \dots, x_n, J) \mapsto (x_1, \dots, x_n, \sum_{i \in J} x_i) .$$

Pour inverser (x_1, \dots, x_n, y) il faut trouver $J \subseteq \{1, \dots, n\}$ tel que $\sum_{i \in J} x_i = y$. Contrairement aux deux problèmes précédents, on sait que ce problème est NP-complet.¹

En *pratique*, on utilise des *constructions spécialisées* de PRG comme RC4, Fortuna, . . . ou alors on utilise des fonctions comme AES qui seront discutées dans le chapitre 7.

4.4 Sommaire

Les notions de PRG et de 1-confidentialité reposent sur la notion d'équivalence calculatoire. On peut adapter le système à masque jetable pour construire un système cryptographique qui assure la 1-confidentialité à partir d'un PRG. L'existence de PRG et de systèmes qui assurent la 1-confidentialité est un problème ouvert. Ce qu'on sait est qu'un PRG existe ssi une fonction à sens unique existe et que l'existence d'une telle fonction implique que $P \subset NP$. Donc, il semble que pour bâtir une fondation solide de la cryptographie il faut commencer par régler le problème ouvert le plus important de l'informatique.

1. Si on trouve un algorithme efficace pour ce problème alors on a un algorithme efficace pour tous les problèmes de la classe NP.

Chapitre 5

Générateurs pseudo-aléatoires de fonctions (*PRF*) et CPA-confidentialité

Dans ce chapitre on introduit la notion de générateur pseudo-aléatoire de *fonction* (*PRF* pour *pseudo-random function generator*). Comme pour la notion de PRG, la notion de PRF repose sur la notion d'équivalence calculatoire. Il est facile de dériver un PRG d'un PRF et il est aussi possible, mais pas aussi simple, de dériver un PRF d'un PRG. Donc, du moins d'un point de vue théorique, on peut voir un PRF comme une abstraction cryptographique qui peut être mise-en-oeuvre avec un PRG. On utilise cette abstraction pour construire des systèmes cryptographiques qui assurent un niveau supérieur de confidentialité qu'on appelle CPA-confidentialité (*CPA* pour *chosen plain-text attack*). Comme le nom le suggère de tels systèmes doivent résister à des attaques sur texte clair choisi. En particulier, on fera l'hypothèse que l'attaquant a accès à la fonction de chiffrement (mais pas à la clef). Le système cryptographique qui assure la CPA-confidentialité à partir d'un PRF utilise à nouveau certaines idées du système à masque jetable. Par ailleurs le système peut être généralisé à des messages composés d'un nombre variable de blocs. Pour ce faire, on va introduire deux nouveaux modes de transmission (après le mode dictionnaire ou *ECB* de la section 1.3) : le mode compteur (*CTR*) et le mode chaîné (*CBC*).

5.1 Fonctions (pseudo-)aléatoires

Une *fonction* de type $[2^n \rightarrow 2^n]$ est spécifiée par un tableau qui associe n bits à chaque entrée spécifiée aussi par n bits. *Tirer une fonction* dans $[2^n \rightarrow 2^n]$ avec probabilité uniforme revient donc à tirer $n \cdot 2^n$ bits de façon indépendante et avec probabilité uniforme. La génération de $n \cdot 2^n$ bits est évidemment un processus dont le coût est exponentiel en n mais souvent on a besoin de connaître la valeur de la fonction seulement dans un nombre réduit de points. Dans ce cas, la génération peut être incrémentale en utilisant une technique de mémorisation. On garde dans une structure de données (par exemple, une table de hachage) les valeurs de la fonction déjà calculées (on les mémorise). Chaque fois qu'on souhaite calculer la fonction dans un point x on vérifie si la valeur de la fonction sur ce point est déjà mémorisée dans la structure. Si c'est le cas, on rend la valeur en question, sinon on génère n bits et on les associe au point x dans la structure.

Un *générateur pseudo-aléatoire de fonctions* (PRF pour *pseudo-random function generator*) est une famille de fonctions :

$$\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n] , \quad n \in \mathbf{N} ,$$

qui associe à n bits une fonction dans $[2^n \rightarrow 2^n]$ qui *semble aléatoire* à un algorithme de décision PPT et telle que le calcul de $\mathcal{F}_n(x)(y)$ est efficace (polynomial en n). La formalisation de cette idée fait face à une difficulté : un algorithme de décision A PPT *n'a pas le temps* d'examiner l'intégralité d'une fonction $f \in [2^n \rightarrow 2^n]$ présentée en tant que tableau. Pour régler cette question on fait appel à la notion d'oracle (voir remarque 25(3)). On suppose que l'algorithme A a la possibilité d'*appeler* une fonction $g : 2^n \rightarrow 2^n$ (coût de chaque appel $O(1)$) un nombre (forcement) polynomial de fois et d'enregistrer la valeur calculée. On dénote un tel algorithme PPT qui utilise g comme un oracle par A^g .

Définition 41 (PRF) *On dit que \mathcal{F} est un PRF si pour tout algorithme de décision PPT A^g (donc qui utilise une fonction $g \in [2^n \rightarrow 2^n]$ comme oracle) on a que la fonction :*

$$\epsilon(n) = |P(X_n = 1) - P(Y_n = 1)|$$

est négligeable où pour $n \in \mathbf{N}$ on définit :

$$X_n = [k \leftarrow 2^n : A^{\mathcal{F}_n(k)}] , \quad Y_n = [f \leftarrow [2^n \rightarrow 2^n] : A^f] .$$

On remarque que l'image des v.a.d. X_n et Y_n est l'ensemble $\{0, 1\}$.

- $P(X_n = 1)$ est la probabilité qu'en tirant k avec probabilité uniforme dans 2^n l'algorithme de décision A retourne 1 en utilisant l'oracle $\mathcal{F}_n(k)$.
- $P(Y_n = 1)$ est la probabilité qu'en tirant une fonction f avec probabilité uniforme dans $[2^n \rightarrow 2^n]$ l'algorithme de décision A retourne 1 en utilisant l'oracle f .

On note qu'un PRF $\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n]$ induit un PRG $\mathcal{G}_n : 2^n \rightarrow 2^{\ell \cdot n}$ en posant :

$$\mathcal{G}_n(k) = \mathcal{F}_n(k)(0)\mathcal{F}_n(k)(1) \cdots \mathcal{F}_n(k)(\ell - 1) ,$$

à condition que $\ell \leq 2^n$. Vice versa on peut construire un PRF à partir d'un PRG qui double la taille :

$$\mathcal{G}_n : 2^n \rightarrow 2^{2^n} .$$

Pour un n fixé, on dénote par $G_0(x)$ ($G_1(x)$) la moitié gauche (droite) de $\mathcal{G}_n(x)$.

Fait 42 *On peut construire un PRF :*

$$\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n] \quad n \in \mathbf{N} ,$$

à partir d'un PRG $\mathcal{G}_n : 2^n \rightarrow 2^{2^n}$ en définissant :

$$\mathcal{F}_n(k)(x_1 \cdots x_n) = G_{x_n}(\cdots (G_{x_1}(k)) \cdots) .$$

Cette construction est due à *Goldreich-Goldwasser-Micali* [GGM86]. En pratique, on utilise des constructions comme AES qui sont efficaces et qui *semblent* se comporter comme un PRF (mais on n'a pas de preuve!) Ce qu'on sait est qu'à partir d'un PRF, on peut construire (de façon efficace) des systèmes qui garantissent un haut niveau de confidentialité et d'intégrité.

5.2 CPA-Confidentialité

La notion de 1-confidentialité (section 4.2) nous a permis d'illustrer comment la notion d'équivalence calculatoire peut être utilisée pour définir la propriété de confidentialité d'un système cryptographique. Cependant, la propriété de 1-confidentialité est trop faible. Par exemple, elle ne dit pas ce qui se passe si le même message est chiffré deux fois. Pour cette raison, on *renforce* la notion de 1-confidentialité en supposant que l'attaquant a *accès à la fonction de chiffrement* (mais pas à la clé). Techniquement, on dénote par $A^{E(k, \cdot)}$ un *algorithme de décision PPT* qui peut appeler la *fonction de chiffrement* $E(k, \cdot)$. Comme dans la définition 33, pour tout $n \in \mathbf{N}$, on a un espace de clefs \mathcal{K}_n , de textes clairs \mathcal{P}_n et de textes chiffrés \mathcal{C}_n avec une fonction G_n pour générer les clefs, une fonction E_n pour chiffrer et une fonction D_n pour déchiffrer.

On définit deux familles de v.a.d. dont l'image varie sur $\{0, 1\}$ et qui dépendent de deux paramètres :

- Un algorithme PPT pour générer pour tout n un couple de textes clairs $m_n, m'_n \in \mathcal{P}_n$ avec $|m_n| = |m'_n|$.
- Un algorithme de décision PPT A^g qui a accès à un oracle g .

Les v.a.d. sont définies de la façon suivante :

$$\begin{aligned} X_n &= [k \leftarrow G_n(1) : A^{E_n(k, \cdot)} \circ (m_n, m'_n, E_n(k, m_n))] \\ Y_n &= [k \leftarrow G_n(1) : A^{E_n(k, \cdot)} \circ (m_n, m'_n, E_n(k, m'_n))] . \end{aligned}$$

La v.a.d X_n diffère de la v.a.d Y_n par le fait que sa troisième composante est un chiffrement de m_n plutôt que de m'_n .

On dit que le système cryptographique assure la CPA-confidentialité si pour toute séquence de textes clairs avec les propriétés précisées ci dessous et pour tout algorithme de décision PPT A^g la fonction :

$$\epsilon(n) = |P(X_n = 1) - P(Y_n = 1)|$$

est négligeable.

Cette définition diffère de la définition 33 par le fait que l'attaquant PPT A a accès à la fonction de chiffrement. Dans la littérature on trouve plusieurs variantes de cette définition. Par exemple [KL07] en propose une basée sur un *jeu* qui s'inspire du jeu qu'on a déjà décrit dans l'exemple 20 en relation avec la notion de distance statistique. Voici une description du jeu :

1. On génère une clef k avec G .
2. L'adversaire PPT, qui utilise E_k comme un oracle, génère deux messages m_0, m_1 de la même taille avec $m_0 \neq m_1$.
3. L'adversaire reçoit un chiffrement $c = E_k(m_b)$ où $b \in \{0, 1\}$ est choisi avec probabilité uniforme.
4. L'adversaire PPT, qui utilise E_k comme un oracle, décide si c est un chiffrement de m_0 ou de m_1 .
5. La probabilité que l'adversaire PPT répond correctement est $1/2$ à une quantité négligeable près (on dit aussi que l'avantage de l'adversaire est négligeable).

Remarque 43 *Pour avoir la CPA-confidentialité, la fonction de chiffrement doit être probabiliste (et la probabilité d'obtenir le même texte en chiffrant 2 fois le même message*

négligeable). En effet, si E_k est déterministe, à partir du triplet (m, m', c) il suffit de calculer $E_k(m)$ et de le comparer à c pour savoir si c est le chiffrement de m ou de m' .¹

On va utiliser un PRF pour concevoir un système cryptographique qui assure la CPA-confidentialité. On considère d'abord le cas où tous les messages ont la même longueur n .

Proposition 44 *Soit $\{\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n]\}_{n \in \mathbf{N}}$ un PRF. Le système suivant assure la CPA-confidentialité. Pour chiffrer un message de taille n on définit :*

$$E_k(m) = [r \leftarrow 2^n : (r, \mathcal{F}_n(k)(r) \oplus m)]$$

et pour déchiffrer (r, c) on définit :

$$D_k(r, c) = \mathcal{F}_n(k)(r) \oplus c .$$

PREUVE. On se limite à présenter les idées principales, voir [KL07] pour une formalisation. D'abord une remarque préliminaire. Si l'on sait que $E_k(m) = (r, c)$ alors on connaît $\mathcal{F}_n(k)(r)$ car :

$$\mathcal{F}_n(k)(r) = c \oplus m .$$

On suppose disposer du triplet :

$$(m, m', (r, c))$$

où l'on sait que (r, c) est un chiffrement de m ou de m' . Alors si on trouve m'' avec un chiffrement (r, c'') on peut calculer $\mathcal{F}_n(k)(r)$ et donc $c \oplus \mathcal{F}_n(k)(r) \in \{m, m'\}$. Donc la probabilité de tirer 2 fois le même r doit être *négligeable*.

La structure de la preuve est la suivante :

- On montre qu'on obtient la *CPA-confidentialité* en supposant que $f \in [2^n \rightarrow 2^n]$ est tirée de *façon aléatoire*.
- Ensuite on remplace f par $\mathcal{F}_n(k)$ est on dérive d'une *attaque à la CPA-confidentialité* une *attaque au PRF* (argument par *réduction*).

L'observation principale est la suivante :

- On suppose que $f : 2^n \rightarrow 2^n$ est choisie avec *probabilité uniforme* et $E_k(m_i) = (r_i, c_i)$ pour $i = 1, \dots, \ell$.
- Si $r_i \neq r_j$ pour $i \neq j$ alors les $f(r_i)$ sont des suites de n bits *indépendants* et tirés avec *probabilité uniforme*.
- On est dans la situation du *masque jetable* où le chiffré ne donne aucune information sur le texte clair.
- Il faut donc se soucier juste du cas où un $r_i = r_j$ avec $i \neq j$ (remarque préliminaire). Mais on peut calculer un chiffrement (=consulter l'oracle) au plus un nombre polynomial $q(n)$ de fois et la probabilité de tirer un r *fixé* est au plus $q(n)/2^n$ qui est *négligeable*. \square

Remarque 45 *Il ne suffit pas d'avoir un PRF pour garantir la CPA-confidentialité, il faut aussi utiliser un schéma correct ! Par exemple, les PRF utilisés en pratique (DES, AES)*

1. Une alternative qui n'est pas considérée dans la suite est d'avoir des fonctions de chiffrement dont la sortie dépend d'un état.

donnent une permutation pseudo-aléatoire avec son inverse. Il est alors tentant d'utiliser les fonctions de chiffrement et déchiffrement suivantes :

$$E_k(m) = \mathcal{F}_n(k)(m) , \quad D_k(c) = (\mathcal{F}_n(k))^{-1}(c) .$$

Mais dans le système dérivé le chiffrement n'est pas probabiliste et donc on ne peut pas garantir la CPA-confidentialité.

Remarque 46 Dans la littérature, on trouve des formalisations de la notion de confidentialité qui sont encore plus restrictives. On notera en particulier la notion de CCA-Confidentialité (CCA pour chosen ciphertext attack) qui permet à l'attaquant un accès limité à la fonction de déchiffrement : à savoir l'attaquant peut déchiffrer autant de messages qu'il le souhaite avant d'essayer de distinguer les chiffrements de deux messages (de même longueur). En pratique, on réalise la notion de CCA-confidentialité en combinant les systèmes qui assurent la CPA-confidentialité avec ceux qui assurent l'intégrité des messages (chapitre 6).

5.3 PRF et mode compteur (CTR)

On s'intéresse maintenant au chiffrement de messages de longueur arbitraire. On suppose avoir un système pour chiffrer des blocs de n bits. Pour chiffrer un message m on le découpe en k blocs de n bits :

$$m = m_1 \cdots m_k .$$

Si $|m|$ est un multiple de n alors on ajoute un bloc m_{k+1} qui contient le mot $10 \cdots 0$ (par exemple). Autrement, on complète le dernier bloc m_k avec un mot de la forme $10 \cdots 0$. Pour retrouver les bits originaux il suffit de regarder à droite du dernier bloc et d'éliminer le mot de la forme $10 \cdots 0$ (cette technique de bourrage est similaire à celle présentée dans la section 1.3).

On suppose que $\{\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n]\}_{n \in \mathbb{N}}$ est un PRF et que après bourrage le message à chiffrer m a la forme $m_1 \cdots m_k$ où chaque bloc est composé de n bits et $1 \leq k < 2^n$. On dénote avec $+$ l'addition modulo 2^n de deux nombres de n bits.

Fait 47 On obtient un système cryptographique qui assure la CPA-confidentialité pour des messages de longueur variable en définissant la fonction de chiffrement par :

$$E_k(m) = [r \leftarrow 2^n : (r, \mathcal{F}_n(k)(r+1) \oplus m_1, \dots, \mathcal{F}_n(k)(r+k) \oplus m_k)]$$

et la fonction de déchiffrement par :

$$D_k(r, c_1, \dots, c_k) = (c_1 \oplus \mathcal{F}_n(k)(r+1), \dots, c_k \oplus \mathcal{F}_n(k)(r+k))$$

Remarque 48 En pratique, il faut veiller à renouveler périodiquement la clé. Par exemple, avec un bloc de taille 64 le nombre de blocs qu'on peut chiffrer est de l'ordre de 2^{30} .

Attaque sur le mode compteur

On considère maintenant une attaque sur le mode compteur. Cette attaque demande un nombre considérable de message chiffrés. Soient :

$$m = m_1 \cdots m_q , \quad m' = m'_1 \cdots m'_q ,$$

et soit (r, c_1, \dots, c_q) le chiffrement de m (ou de m'). On chiffre ℓ messages composés de q blocs.

$$E_k(m_t) = (r_t, c_{t,1}, \dots, c_{t,q}) , \quad t = 1, \dots, \ell .$$

On reçoit un chiffré avec un r tel que, par exemple, $r + i = r_t$ et $0 \leq i < q$ (le segment $[r, r + q - 1]$ intersecte le segment $[r_t, r_t + q + 1]$). Alors :

$$\begin{aligned} c_{i+1} &= \mathcal{F}_n(k)(r + i + 1) \oplus m_{i+1} \\ c_{t,1} &= \mathcal{F}_n(k)(r_t + 1) \oplus m_{t,1} = \mathcal{F}_m(k)(r + i + 1) \oplus m_{t,1} \\ \mathcal{F}_m(k)(r_t + 1) &= \mathcal{F}_n(k)(r + i + 1) = c_{t,1} \oplus m_{t,1} \\ m_{i+1} &= c_{i+1} \oplus \mathcal{F}_n(k)(r + i + 1) . \end{aligned}$$

Il faut donc calculer la probabilité que deux segments de longueur q dans un intervalle de longueur 2^n s'intersectent. Cette probabilité est bornée par $2q/2^n$. Si on chiffre ℓ messages on a ℓ segments de longueur q . La probabilité que le chiffrement d'un message intersecte un de ces segments est bornée par $\ell \cdot 2q/2^n$.

5.4 Mode chaîné (CBC)

On présente un mode chaîné (*CBC* pour *cypherblock chaining*) pour traiter des messages composés de plusieurs blocs. Ce mode peut être utilisé à la fois pour chiffrer un message et pour calculer son *MAC* (ou code d'authentification, voir chapitre 6). Le mode *CBC* utilise l'opération d'ou exclusif (*xor*). Pour cette raison, on suppose que les textes clairs et chiffrés utilisent un alphabet binaire $\Sigma = \{0, 1\} = 2$. Le calcul dépend d'un vecteur d'initialisation $IV \in 2^n$. Ce vecteur peut être *public*. Dans certains cas, IV peut être fixé une fois pour toutes. Dans d'autres situations, il doit être choisi de façon aléatoire à chaque utilisation du mode *CBC*.

Soit k une clef et E_k une permutation sur 2^n . A partir d'un message m composé de t blocs de n bits $m = m_1, \dots, m_t$ on définit une suite de $t + 1$ blocs c_0, c_1, \dots, c_t par :

$$\begin{aligned} c_0 &= IV , \\ c_{j+1} &= E_k(c_j \oplus m_{j+1}) , \quad j = 0, \dots, t - 1 . \end{aligned}$$

Si D_k est l'inverse de la fonction E_k il est possible d'inverser le processus en effectuant le calcul suivant :

$$\begin{aligned} n_0 &= IV , \\ n_{j+1} &= D_k(c_{j+1}) \oplus c_j , \quad j = 0, \dots, t - 1 . \end{aligned}$$

On a $n_j = m_j$, car :

$$n_{j+1} = D_k(E_k(m_{j+1} \oplus c_j)) \oplus c_j = m_{j+1} \oplus c_j \oplus c_j = m_{j+1} .$$

5.5 Confidentialité CPA en mode CBC

Fait 49 Si le PRF génère une permutation pseudo-aléatoire (PRP) avec son inverse alors on peut utiliser le mode CBC pour obtenir un système avec confidentialité CPA.

$$\begin{aligned} E_k(m_1 \dots m_\ell) &= [r \leftarrow 2^n; c_0 = r; c_1 = \mathcal{F}_n(k)(c_0 \oplus m_1); \dots; \\ &\quad c_\ell = \mathcal{F}_n(k)(c_{\ell-1} \oplus m_\ell) : (c_0, c_1, \dots, c_\ell)] \end{aligned}$$

$$\begin{aligned} D_k(c_0, c_1, \dots, c_\ell) &= [m_1 = c_0 \oplus (\mathcal{F}_n(k))^{-1}(c_1); \dots; \\ &\quad m_\ell = c_{\ell-1} \oplus (\mathcal{F}_n(k))^{-1}(c_\ell) : (m_1, \dots, m_\ell)] . \end{aligned}$$

Remarque 50 On suppose que l'attaquant a une façon pour prédire le prochain vecteur d'initialisation r' utilisé dans un chiffrement en mode CBC. On montre que dans ce cas il peut proposer deux messages m et m' et deviner si le chiffrement correspond au chiffrement de m ou de m' . On dénote par $\mathbf{0}$ un bloc composés de 0. L'attaquant commence par calculer le chiffrement du bloc $\mathbf{0}$, à savoir :

$$E_k(\mathbf{0}) = (r, \mathcal{F}(k)(\mathbf{0} \oplus r)) = (r, \mathcal{F}(k)(r)) .$$

Ainsi il apprend la valeur de $\mathcal{F}(k)$ sur r . Maintenant, il prédit que le prochain vecteur d'initialisation utilisé sera r' et propose deux messages : un message $m = r \oplus r'$ et un message $m' \neq m$ choisi avec probabilité uniforme. Ensuite il obtient un chiffrement soit de m soit de m' qui a la forme (r', c) . L'attaquant a une façon pour distinguer ces deux situations car s'il s'agit d'un chiffrement du message m alors on doit avoir :

$$\mathcal{F}(k)(r' \oplus (r \oplus r')) = \mathcal{F}(k)(r) = c ,$$

d'autre part avec une haute probabilité $(r' \oplus m') \neq r$ et $\mathcal{F}(k)(r' \oplus m') \neq c$.

5.6 Schéma de Feistel

On vient de voir que pour le mode CBC on a besoin de générer une *permutation* pseudo-aléatoire. Le schéma de Feistel permet de transformer un générateur de fonctions en un générateur de permutations et sous certaines conditions [LR88] on peut montrer que si le générateur de fonctions est un PRF alors le générateur de permutations est un PRP.

Soit $\Sigma = \{0, 1\}$ un alphabet binaire. On fixe :

- Une méthode pour associer à une clef k , une fonction $f_k : 2^t \rightarrow 2^t$. A noter que f_k n'est pas forcément injective !
- Un nombre de tours r .
- Une méthode pour générer d'une clef k , r clefs de tour k_1, \dots, k_r .

Le schéma de Feistel est une fonction de chiffrement (et déchiffrement) qui opère sur des blocs de longueur $2t$ en r tours.

Chiffrement

- On divise le texte clair p en deux moitiés $L_0 R_0$.
- On définit pour $i = 1, \dots, r$:

$$L_i R_i = R_{i-1} (L_{i-1} \oplus f_{k_i}(R_{i-1})) .$$

- On pose :

$$E_k(p) = L_r R_r .$$

Déchiffrement

- Pour $L_i R_i = R_{i-1} (L_{i-1} \oplus f_{k_i}(R_{i-1}))$ on dérive :

$$\begin{aligned} R_{i-1} L_{i-1} &= L_i (R_i \oplus f_{k_i}(R_{i-1})) \\ &= L_i (R_i \oplus f_{k_i}(L_i)) . \end{aligned}$$

- Étant donné un bloc de texte chiffré c , on le divise en $L_r R_r$ et on calcule $L_i R_i$ pour $i = r - 1, \dots, 0$ en utilisant les clefs k_r, \dots, k_1 .

— Alors :

$$D_k(c) = L_0 R_0 .$$

Remarque 51 *On peut déchiffrer même si les fonctions f_{k_i} ne sont pas injectives ; dans le déchiffrement on n'a pas à calculer l'inverse de f_{k_i} !*

5.7 Sommaire

Un PRF est une famille de fonctions $\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n]$ telle qu'un adversaire PPT ne peut pas distinguer avec probabilité non-négligeable une fonction générée d'une fonction tirée avec probabilité uniforme. On peut dériver un PRG d'un PRF et construire un PRF à partir d'un PRG. Un système assure la CPA-confidentialité si un adversaire PPT qui a accès à la fonction de chiffrement n'arrive pas à distinguer les chiffrements de deux messages de même longueur. On utilise un PRF pour construire des systèmes cryptographiques qui assurent la CPA-confidentialité. En particulier, pour traiter des messages composés d'un nombre variable de blocs on peut utiliser le mode compteur (*CTR*) ou le mode chaîné (*CBC*).

Chapitre 6

Intégrité

On considère le problème suivant : Bob connaît la signature d'Alice. S'il reçoit une lettre avec la signature d'Alice il est confiant que la lettre a été rédigée par Alice. On souhaite avoir un mécanisme similaire pour les documents électroniques, par exemple les contrats électroniques, les transactions bancaires, ... A noter que la signature doit permettre de vérifier à la fois l'*origine* du message et son *intégrité*. Il y a *deux mécanismes principaux* :

- Les *codes d'authentification de messages* (*message authentication codes, MAC*) pour les systèmes *symétriques*.
- Les *signatures digitales* pour les systèmes *asymétriques*.

Dans ce chapitre on s'intéresse au premier mécanisme.

6.1 Codes d'authentification de messages (MAC)

Très souvent, les systèmes qui assurent la *confidentialité* d'un message ne garantissent pas son *intégrité*. Par exemple, le *masque jetable* (section 2.1) garantit une confidentialité parfaite mais pour compléter les bits du *message reçu* il suffit de compléter les bits du *message chiffré*. En effet, si $x \oplus k = c$ alors $\bar{c} \oplus k = \bar{x}$.

Pour un autre exemple, on considère un message composé de plusieurs blocs : $m = m_1 \cdots m_\ell$. Dans ce cas, le chiffrement du message, basé sur un PRP et en mode CBC (voir section 5.4), a la forme :

$$(c_0 = r, c_1 = \mathcal{F}(k)(c_0 \oplus m_1), \dots, c_\ell = \mathcal{F}(k)(c_{\ell-1} \oplus m_\ell)) .$$

Si l'on *remplace* r par r' le récepteur obtient :

$$m'_1 = (\mathcal{F}(k))^{-1}(c_1) \oplus r', m_2 = (\mathcal{F}(k))^{-1}(c_2) \oplus c_1, \dots m_\ell = (\mathcal{F}(k))^{-1}(c_\ell) \oplus c_{\ell-1} .$$

Donc l'*intégrité du premier bloc est compromise*.

En général, si toute suite de bits d'une certaine longueur est *admissible*, on peut juste générer une suite de bits et prétendre que c'est un texte chiffré.

Pour traiter la propriété d'intégrité, on suit un chemin similaire à celui parcouru pour la propriété de confidentialité, à savoir on définit quand un système assure l'intégrité et ensuite on montre comment construire un tel système en supposant l'existence d'un PRF.

Pour garantir l'intégrité, l'idée de base est d'attacher à un message une étiquette (ou *tag*). Dans le cadre de la cryptographie symétrique, l'émetteur et le récepteur partagent une clef

secrète. A partir du message et de la clef l'émetteur construit une étiquette et à partir du message, de l'étiquette et de la clef le récepteur vérifie l'intégrité du message.

Définition 52 (MAC) Soient \mathcal{P} , \mathcal{T} et \mathcal{K} des ensembles finis qui représentent respectivement les textes clairs, les étiquettes (ou MAC ou tags) et les clefs. Un système qui assure l'intégrité est composé de trois algorithmes $PPT : G, Mac$ et V avec les types et les fonctionnalités suivantes :

$$\begin{aligned} G : 1 &\rightarrow \mathcal{K} && (\text{générateur de clef}) \\ Mac : \mathcal{K} \times \mathcal{P} &\rightarrow \mathcal{T} && (\text{générateur MAC}) \\ V : \mathcal{K} \times \mathcal{P} \times \mathcal{T} &\rightarrow \{0, 1\} && (\text{vérificateur MAC}) \end{aligned}$$

tels que :

$$\forall k, m, t \quad V(k, m, t) = 1 \text{ ssi } t = Mac(k, m) .$$

De plus, le système doit satisfaire la propriété suivante : tout attaquant PPT qui connaît les MAC d'un nombre (polynomial) de messages m_1, \dots, m_ℓ a une probabilité négligeable de produire un MAC valide pour un message m différent de m_1, \dots, m_ℓ .

Remarque 53 Pour formaliser la définition, on peut introduire une fonction :

$$M_{k,m} : \mathcal{K} \times \mathcal{P} \rightarrow \mathcal{T} ,$$

telle que $M_{k,m}(m)$ est une étiquette choisie avec probabilité uniforme et $M_{k,m}(m') = Mac(k, m')$ si $m \neq m'$. Ensuite on suppose que l'attaquant PPT a accès à la fonction $M_{k,m}$ en tant que oracle (comme dans la définition de CPA-confidentialité) et que son but est de produire une étiquette valide pour m .

6.2 CBC-MAC

Si un PRF existe alors on peut construire un MAC sûr (en pratique, on peut utiliser AES). Une construction possible est d'adapter le mode de transmission CBC :

1. A partir d'une clef k on dérive deux clefs k_1 et k_2 .
2. On décompose le message m en ℓ blocs $m = m_1 \cdots m_\ell$ et on calcule le dernier bloc du mode CBC par rapport à la clef k_1 :

$$\begin{aligned} c_0 &= 0 && (\text{le vecteur d'initialisation est fixé!}) \\ c_{j+1} &= \mathcal{F}(k_1)(c_j \oplus m_{j+1}) && j = 0, \dots, \ell - 1 . \end{aligned}$$

3. Ensuite on chiffre c_ℓ (le dernier bloc) avec la deuxième clef :

$$Mac(k, m) = \mathcal{F}(k_2)(c_\ell) .$$

Pour vérifier, il suffit de refaire le calcul et de comparer le résultat au MAC associé au message. Notez que dans la vérification on n'a pas besoin de calculer l'inverse de $\mathcal{F}(k)$. Pour cette raison, CBC-MAC peut utiliser un PRF (et non pas un PRP comme dans le schéma pour la confidentialité CPA décrit dans la section 5.4).

Il y a aussi d'autres schémas pour construire un MAC. Par exemple, *HMAC* et *NMAC* qui utilisent les fonctions de hachage.¹ Dans toutes ces méthodes, les fonctions *Mac* qui génèrent les codes d'authentification ne sont pas probabilistes (contrairement aux fonctions de chiffrement).

1. Ces fonctions ont des propriétés similaires aux fonctions à sens unique et résistantes aux collisions décrites dans la section 1.6

Attaque sur CBC-MAC simplifié

L'utilisation de deux clefs dans le calcul du *Mac* sert à éviter l'attaque suivante.

- On suppose qu'on utilise un *CBC-MAC faible* qui utilise une seule clef (dans le calcul du MAC ci dessus, l'étape 3 est omise).
- On suppose aussi que $m = m_1 \cdots m_\ell$, $\ell \geq 2$ et qu'on connaît les CBC-MAC (faibles) $c_{\ell-1}$ de $m_1 \cdots m_{\ell-1}$ et c_ℓ de $m_1 \cdots m_\ell$.
- On peut alors construire le message suivant composé de $\ell + 1$ blocs :

$$m' = m \cdot (m_\ell \oplus c_\ell \oplus c_{\ell-1}) .$$

- Et on remarque qu'on sait calculer son CBC-MAC (faible) :

$$\mathcal{F}(k_1)(c_\ell \oplus m_\ell \oplus c_\ell \oplus c_{\ell-1}) = \mathcal{F}(k_1)(m_\ell \oplus c_{\ell-1}) = c_\ell .$$

Piège (CBC-MAC)

On a vu qu'un PRP en mode CBC *n'assure pas* la *CPA-confidentialité* si le vecteur d'initialisation est *prévisible* (section 5.5). On pourrait donc penser que dans la mise-en-oeuvre du CBC-MAC il convient de choisir le vecteur d'initialisation de façon aléatoire. Or il se trouve qu'avec un *choix aléatoire*, CBC-MAC *n'assure pas l'authentification*! Cet exemple illustre bien la nécessité de *prouver* la sécurité d'un schéma (même si très souvent en cryptographie la preuve repose sur certaines conditions). Voici une attaque sur CBC-MAC avec choix aléatoire du vecteur d'initialisation.

- D'abord, on remarque que si le vecteur d'initialisation est aléatoire alors le MAC doit *inclure* ce vecteur.
- Pour des messages composés d'un *bloc*, le MAC prend la forme suivante :

$$\text{Mac}(m, k) = (r, \mathcal{F}(k)(r \oplus m)) .$$

- Alors : $(\bar{r}, \mathcal{F}(k)(r \oplus m))$ est un *MAC valide* pour \bar{m} car : $\bar{r} \oplus \bar{m} = r \oplus m$.

6.3 Combiner confidentialité et intégrité

On peut imaginer différentes façons de *combiner* un schéma qui assure la *confidentialité* avec un *MAC* :

1. Chiffrement suivi par MAC : $[c = E(k, m) : (c, \text{Mac}(k', c))]$.
2. Chiffrement et MAC : $(E(k, m), \text{Mac}(k', m))$.
3. MAC suivi par chiffrement : $E(k, (m, \text{Mac}(k', m)))$.

Le premier schéma (à savoir chiffrement suivi par MAC) est le plus *robuste*. Par exemple, dans le deuxième schéma la valeur $\text{Mac}(k', m)$ pourrait dévoiler des informations sur le texte clair m . En tout cas, une règle générale est qu'il ne faut *pas* utiliser la même clef pour le chiffrement et le MAC (donc $k \neq k'$). Aussi une bonne pratique d'ingénierie est de construire des systèmes dans lesquels on assure à la fois la *confidentialité* et l'*intégrité* des messages échangés.

6.4 Sommaire

A partir d'un *PRF* \mathcal{F} , on peut construire un MAC qui assure la propriété d'intégrité de la façon suivante où l'on dénote par *ext* une fonction qui extrait d'une clef un couple de clefs :

$$\begin{aligned} \text{Mac}(k, m_1 \cdots m_\ell) &= [(k_1, k_2) = \text{ext}(k); c_0 = 0; c_1 = \mathcal{F}(k_1)(c_0 \oplus m_1); \dots; \\ &\quad c_\ell = \mathcal{F}(k_1)(c_{\ell-1} \oplus m_\ell) : \mathcal{F}(k_2)(c_\ell)] . \end{aligned}$$

Pour des textes composés d'un nombre fixe de blocs l'utilisation de deux clefs différentes (k_1 et k_2) est redondante.

C'est une bonne pratique de construire des systèmes qui assurent la confidentialité et l'intégrité. Une façon robuste de combiner les systèmes pour chiffrer et pour authentifier est de chiffrer d'abord et ensuite de calculer le Mac du message chiffré en utilisant deux clefs différentes. On trouve dans la littérature des définitions de confidentialité CCA (*chosen ciphertext attack*) qui sont encore plus restrictives que la confidentialité CPA et qui imposent l'utilisation de mécanismes pour garantir l'intégrité.

Chapitre 7

Permutations pseudo-aléatoires : pratique

Parmi les fonctions $f : 2^n \rightarrow 2^n$ il y a $(2^n)!$ permutations. Une notation pour représenter une permutation aléatoire a besoin de $\log_2((2^n)!) \approx n2^n$ bits. Un générateur de permutations pseudo-aléatoires *pratique* ne génère qu'un petit nombre de permutations parmi celles possibles. Par exemple, 2^n permutations parmi les $(2^n)!$ possibles. Dans ce cas, les permutations générées peuvent être représentées de façon compacte.

Les définitions de PRG et PRF suggèrent un standard avec lequel mesurer les réalisations pratiques de générateurs pseudo-aléatoires de permutations (PRP). Le *critère idéal* est qu'il doit être difficile de distinguer une permutation aléatoire d'une permutation générée par la méthode. Plus pratiquement, la *recherche de la clé* (=germe) de la permutation pseudo-aléatoire doit essentiellement demander une recherche exhaustive.

Il est possible de construire des PRP dont la sécurité repose sur la difficulté de problèmes bien étudiés comme le problème de la factorisation du produit de deux grands nombres premiers. Malheureusement, ces constructions ne sont pas aussi efficaces que celles qui sont utilisées en pratique comme DES et AES et qui sont basées sur un mélange savant de quelques fonctions simples à calculer (permutations, inversions,...). L'évaluation de la sécurité de ces générateurs est *empirique* : on montre que les meilleures méthodes d'analyse *connues* ne marchent pas. En d'autres termes, toute attaque (connue) va prendre un temps proportionnel à 2^n où n est la taille de la clef. Dans ce chapitre, on se limite à donner une description approximative de la structure de DES et AES. Le livre [DR02] contient une discussion approfondie de la sécurité (empirique) d'AES.

7.1 DES

DES (*Data encryption standard*) est un système formalisé autour de 1977 qui utilise le schéma de Feistel (section 5.6). Soient $\mathcal{P} = \mathcal{C} = 2^{64}$ et soit $\mathcal{K} = 2^{56}$. Une clef est complétée à 64 bits en insérant 8 bits aux positions : 8, 16, 24, 32, 40, 48, 56, 64. Ces bits sont utilisés pour détecter et corriger 1 erreur. Dans DES, on utilise $r = 16$ tours. On pose $t = 64/2 = 32$ bits. Les clefs de tour sont sur 48 bits.

Fonction d'expansion

DES utilise une fonction d'expansion $Ex : 2^{32} \rightarrow 2^{48}$ qui est spécifiée par le tableau suivant. On remarquera qu'en première et dernière colonne on trouve des bits de l'entrée qui paraissent aussi dans les colonnes centrales.

| | | | | | |
|----|----|----|----|----|----|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

S-boîtes

DES utilise aussi une série de *S-boîtes* (S-boxes) qui sont des fonctions (non-affines) $S_i : 2^6 \rightarrow 2^4$, $i = 1, \dots, 8$. Par exemple, la boîte S_1 est spécifiée par le tableau suivant :

| | [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] | [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|
| [0] | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| [1] | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| [2] | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| [3] | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

On remarquera que les valeurs dans chaque ligne sont *différentes*. Le calcul de la sortie d'une S-boîte se passe de la façon suivante. Soit l'entrée $B = b_1b_2b_3b_4b_5b_6$. Alors :

- $i = (b_1b_6)_2 \in \{0, 1, 2, 3\}$ détermine l'indice de la ligne.
- $j = (b_2b_3b_4b_5)_2 \in \{0, \dots, 15\}$ détermine l'indice de la colonne.

Le résultat est la valeur de coordonnées (i, j) exprimée en base 2. Par exemple, pour déterminer $S_1(001011)$ on calcule :

$$i = (01)_2 = 1, \quad j = (0101)_2 = 5, \quad S_1[1, 5] = 2, \quad S_1(001011) = 0010.$$

Remarque 54 La composition de l'expansion et des S-boîtes est bien une permutation sur 2^{32} . En effet, si on applique l'expansion le 1er et 6ème bits sont redondants. Ensuite on applique un xor qui est une fonction injective. Enfin la S-boîte va utiliser le 1er et 6ème bit pour associer de façon injective 4 bits aux 4 bits centraux.

Calcul de $f_{k_i}(R_{i-1})$

$$R' := Ex(R_{i-1}) \quad (\text{Expansion } R_i \text{ à 48 bits})$$

$$B_1 \cdots B_8 := R' \oplus k_i \quad (B_i \text{ a 6 bits})$$

$$C_i := S_i(B_i), i = 1, \dots, 8 \quad (C_i \text{ a 4 bits})$$

$$P(C_1 \cdots C_8) \quad (P \text{ pour une permutation sur 32 bits})$$

A noter qu'avant de commencer les 16 tours, DES applique une permutation initiale à ne pas confondre avec la permutation P qui est spécifiée par le tableau suivant :

| | | | |
|----|----|----|----|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

Génération de clefs de tour

On dispose des fonctions suivantes :

$$f_1 : 2^{64} \rightarrow 2^{28} \times 2^{28} , \quad f_2 : 2^{28} \times 2^{28} \rightarrow 2^{48} ,$$

et des fonctions de décalage :

$$\text{shift}_i \text{ pour } i = 1, \dots, 16 .$$

Alors, étant donné une clef k , on calcule :

$$\begin{aligned} C_0 D_0 &= f_1(k), \\ C_i &= \text{shift}_i(C_{i-1}), \quad D_i = \text{shift}_i(D_{i-1}), \quad i = 1, \dots, 16, \\ k_i &= f_2(C_i D_i) . \end{aligned}$$

Remarque 55 *Permutation, expansion, S-boîtes, générateurs de clefs sont spécifiées par certaines tables fixées. Si on veut implémenter DES, il faut faire référence à ces tables, autrement, l'intérêt de ces tables est limité car on ne dispose pas d'une théorie qui justifie clairement leur définition.*

Triple DES

L'espace des clefs dans DES est relativement petit (2^{56} clefs possibles). Pour améliorer la sécurité du système, une première approche pourrait être de considérer un *chiffrement double*. Une clef est un couple de clefs (k_1, k_2) . Pour chiffrer, on calcule :

$$E_{(k_1, k_2)}(m) = E_{k_2}(E_{k_1}(m)) ,$$

et pour déchiffrer, on calcule :

$$D_{(k_1, k_2)}(c) = D_{k_1}(D_{k_2}(c)) .$$

Pour que cette approche ait une chance de marcher, il faut au moins supposer qu'étant données les clefs k_1, k_2 , il est 'improbable' qu'il y ait une clef k telle que : $E_{k_2} \circ E_{k_1} = E_k$. Cette propriété est fausse pour les fonctions affines (voir exercice D.2), mais peut être vérifiée par d'autres classes de fonctions. Par exemple, si les fonctions E_{k_1} et E_{k_2} sont des polynômes de degré 2 il est improbable que leur composition soit un polynôme de degré 2. Avec cette hypothèse, on pourrait espérer qu'une attaque par recherche complète contre le système à

chiffrement double demanderait 2^{2n} essais . . . Malheureusement, une *attaque au milieu* (*meet in the middle attack*) est possible. En effet, en connaissant un certain nombre de couples texte clair-texte chiffré, on peut réduire l'espace de recherche. On suppose connaître :

$$c_i = E_{k_2}(E_{k_1}(p_i)), \quad i = 1, 2 .$$

Alors, il faut que :

$$E_{k_1}(p_i) = D_{k_2}(c_i), \quad i = 1, 2 .$$

Pour toutes les clefs k , on calcule (on a besoin d'une mémoire de taille 2^n) :

$$E_k(p_1) .$$

Pour toutes les clefs possibles k' , on calcule :

$$D_{k'}(c_1) ,$$

et on vérifie si $E_k(p_1) = D_{k'}(c_1)$. Alors on peut tomber sur $k = k_1$ et $k' = k_2$. Pour améliorer la confiance dans le résultat, on vérifie :

$$E_k(p_2) = D_{k'}(c_2) .$$

Évidemment, le plus de couples texte clair-texte chiffré le mieux.

A cause de l'attaque au milieu, le chiffrement double n'est pas utilisé. Une approche différente qui évite l'attaque décrite est de *chiffrer-déchiffrer-chiffrer*. On définit :

$$c = E_{k_1}(D_{k_2}(E_{k_1}(p))) , \quad p = D_{k_1}(E_{k_2}(D_{k_1}(c))) .$$

Ceci est connu comme *chiffrement triple* et il s'agit de la méthode standard pour 'améliorer' la sécurité de DES sans changer l'algorithme. A noter que si $k_1 = k_2$ alors le chiffrement triple est la même chose que le chiffrement standard.

7.2 Un exemple de réseau de substitution permutation

En pratique, le chiffrement triple n'est pas très efficace et ceci a motivé la recherche d'un nouveau standard qui a abouti avec la proposition du standard AES (*Advanced Encryption Standard*). Le standard a été spécifié en 1997. Il prévoit des blocs de 128 bits, avec des clefs de 128, 192 ou 256 bits, et avec des contraintes en termes de vitesse, mémoire et simplicité de mise en oeuvre (notamment par des circuits dédiés). Une mise en oeuvre de ce standard a été sélectionnée en 2000 dans une compétition internationale ouverte par un processus de confrontation et d'élimination.

AES utilise (entre autres) l'idée de réseau de substitution-permutation dont on va donner un exemple à petite échelle. On veut définir une fonction de chiffrement à bloc $E_k : 2^{16} \rightarrow 2^{16}$.

Substitution On utilise la première ligne de la première boîte de DES (section 7.1) qui est une fonction $S : 2^4 \rightarrow 2^4$ spécifiée par le tableau suivant :

| | | | | | | | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Entrée : | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| Sortie : | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |

Permutation Comme permutation on utilise une fonction $P : \{1, \dots, 16\} \rightarrow \{1, \dots, 16\}$ qui est spécifiée par le tableau suivant :

| | | | | | | | | | | | | | | | | |
|----------|---|---|---|----|---|---|----|----|---|----|----|----|----|----|----|----|
| Entrée : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Sortie : | 1 | 5 | 9 | 13 | 2 | 6 | 10 | 14 | 3 | 7 | 11 | 15 | 4 | 8 | 12 | 16 |

Clef Une clef K est aussi sur 16 bits. La clef K_i du tour i pour $i = 1, 2, 3, 4$ est obtenue en effectuant $4 * (i - 1)$ décalages circulaires vers la droite de la clef K . Ainsi si $K = x_1 \cdot x_2 \cdot x_3 \cdot x_4$ et $x_i \in 2^4$ pour $i = 1, 2, 3, 4$, on aura :

$$\begin{aligned} K_1 &= x_1 \cdot x_2 \cdot x_3 \cdot x_4 \\ K_2 &= x_4 \cdot x_1 \cdot x_2 \cdot x_3 \\ K_3 &= x_3 \cdot x_4 \cdot x_1 \cdot x_2 \\ K_4 &= x_2 \cdot x_3 \cdot x_4 \cdot x_1 \end{aligned}$$

Tour Le chiffrement se décompose en 4 tours. Le tour i , pour $i = 1, 2, 3, 4$, consiste à transformer un bloc $x = (x_1 \cdot x_2 \cdot x_3 \cdot x_4)$, où $x_i \in 2^4$ comme suit :

$$\begin{aligned} (y_1 \cdot y_2 \cdot y_3 \cdot y_4) &= (x \oplus K_i) && \text{(XOR avec la clef)} \\ (w_1 \cdot w_2 \cdot w_3 \cdot w_4) &= (S(y_1) \cdot S(y_2) \cdot S(y_3) \cdot S(y_4)) && \text{(Substitution)} \\ z &= P(w_1 \cdot w_2 \cdot w_3 \cdot w_4) && \text{(Permutation)} \end{aligned}$$

Exercice 56 Expliquer comment déchiffrer.

7.3 Structure d'AES

On décrit la structure d'AES en ignorant un certain nombre de détails et en se limitant au cas où la clef a 128 bits.

Comme dans les cas déjà étudiés, AES contient une méthode pour générer des clefs de tour. La fonction qui génère ces clefs comporte une composante non-linéaire. A la différence de DES, AES n'utilise pas le schéma de Feistel et il manipule les données au niveau des octets. Les transformations sont donc définies sur 16 octets (soit 128 bits).

AES comporte 10 tours et chaque tour se décompose en 3 phases.

- Application d'une substitution S à chaque octet. La substitution est une fonction non-linéaire $S : 2^8 \rightarrow 2^8$. AES utilise une seule substitution et non pas 8 comme DES.
- Application de fonctions linéaires pour *diffuser* les données. Dans cette phase on peut voir les 16 octets comme une matrice 4×4 avec éléments dans 2^8 . La diffusion consiste d'abord à décaler de façon circulaire la i -ème ligne de la matrice de i positions vers la gauche, pour $i = 0, 1, 2, 3$. Ensuite chaque colonne de la matrice obtenue est multipliée par une matrice 4×4 prédéfinie.
- Combinaison des données avec la clef de tour à l'aide d'un ou exclusif.

Comme nous l'avons déjà dit, dans AES les calculs se déroulent au niveau des octets. D'un point de vue conceptuel, on peut expliquer ces calculs comme des opérations arithmétiques sur un *corps fini* dont les éléments sont en correspondance bijective avec les octets ; le corps en question contient donc exactement 2^8 éléments. On décrit brièvement la structure algébrique de ce corps.¹

Parmi les propriétés remarquables des corps finis, on a les suivantes :

1. La théorie des corps finis est riche en résultats et en applications (voir, par exemple, [McE87]).

- un corps fini doit avoir un nombre d'éléments égal à p^n où p est un nombre premier et $n \geq 1$.
- pour chaque nombre de la forme p^n , p premier, $n \geq 1$, il existe à isomorphisme près un seul corps avec p^n éléments.

On montre en appendice (voir corollaire 93), que pour tout nombre premier p , les entiers modulo p forment un *corps* avec comme opérations l'addition et la multiplication modulaire. En particulier, $\mathbf{Z}_2 = \{0, 1\}$ est le plus petit corps. Dans ce corps, l'opération d'addition correspond à l'ou exclusif et l'opération de multiplication à la conjonction.

Il reste donc à voir comment construire un corps avec p^n éléments si p est premier et $n > 1$. Cette question a une réponse assez simple : il suffit de considérer les polynômes de degré au plus $n - 1$ et à coefficients dans \mathbf{Z}_p . Dans le cas qui nous intéresse, $p = 2$ et $n = 8$. Un élément du corps est donc un polynôme de la forme :

$$a_7 \cdot x^7 + \cdots + a_1 \cdot x + a_0 \quad (7.1)$$

où les coefficients a_i varient sur $\mathbf{Z}_2 = \{0, 1\}$ et les opérations d'addition et de multiplication (+ et \cdot) sont celles de \mathbf{Z}_2 .

Comment additionner et multiplier deux polynômes ? Il se trouve que pour additionner deux polynômes il suffit d'additionner les coefficients modulo 2 ou, de façon équivalente, de calculer les ou exclusif des coefficients. L'opération de multiplication de polynômes s'inspire aussi de la multiplication dans \mathbf{Z}_p . Pour calculer le produit de deux nombres dans \mathbf{Z}_p , on calcule le produit ordinaire et on retient le reste de la division par p . De la même façon, pour calculer le produit de deux polynômes de degré au plus $n - 1$ et à coefficients dans \mathbf{Z}_p , on commence par calculer le produit des polynômes et ensuite on retient le reste de la division du polynôme obtenu par un polynôme dit *irréductible*. Un polynôme irréductible est un polynôme qui ne s'exprime pas comme le produit de deux polynômes non triviaux. De ce point de vue, la définition d'un polynôme irréductible rappelle celle d'un nombre premier. Dans le cas d'AES, le polynôme irréductible choisi est : $p(x) = x^8 + x^4 + x^3 + x + 1$. Donc pour multiplier deux polynômes de la forme (7.1), on commence par calculer un polynôme de degré au plus 14 et ensuite on retient le reste de la division par $p(x)$. En pratique, dans la mise-en-oeuvre d'AES, on precalcule des tables pour multiplier et pour calculer l'inverse multiplicative.

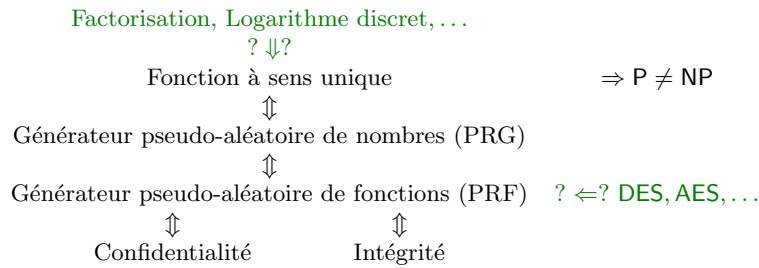
7.4 Sommaire

La pratique de la construction de générateurs aléatoires de permutations (PRP) tels que DES et AES s'appuie sur un certain nombre d'heuristiques : application d'une opération non-linéaire, permutation pour diffuser les données, combinaison avec une clef de tour,...

Parmi les techniques d'attaque développées pour tester ces systèmes, on distingue les techniques dites *linéaires* qui utilisent une approximation linéaire des fonctions non-linéaires utilisées dans le chiffrement (par exemple, les S-boîtes de DES) et les techniques dites *différentielles* qui comparent l'entrée et la sortie de la fonction de chiffrement. Pour DES et AES, ces techniques semblent marcher seulement dans des situations simplifiées (par exemple, avec un nombre de tours réduit).

Sommaire : cryptographie symétrique

Le diagramme suivant résume la situation pour la cryptographie symétrique. Les implications $? \Rightarrow ?$ sont des *voeux pieux*. Il n’y a pas de preuve et on ne semble pas près d’en avoir une !



La table 7.1 résume les constructions présentées. On trouve dans la littérature des constructions de PRG et PRF dont la sécurité repose sur certains problèmes difficiles de l’arithmétique modulaire. Par exemple, le PRG dans [BBS86] et le PRF dans [NR04]. L’état de l’art est que ces constructions ‘prouvées’ sont considérablement moins efficaces que celles, par exemple, basées sur AES et pour cette raison elles ne sont pas utilisées en pratique.

1-confidentialité : à partir d’un **PRG** $\mathcal{G}_n : 2^n \rightarrow 2^{\ell(n)}$:

$$E_k(m) = \mathcal{G}_n(k) \oplus m$$

CPA-confidentialité : à partir d’un **PRF** $\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n]$ en **mode CTR** :

$$E_k(m_1 \cdots m_\ell) = [r \leftarrow 2^n : (r, \mathcal{F}(k)(r+1) \oplus m_1, \dots, \mathcal{F}(k)(r+\ell) \oplus m_\ell)]$$

Variante possible avec **PRP** et **mode CBC** (voir section 5.5).

Intégrité : à partir d’un **PRF** $\mathcal{F}_n : 2^n \rightarrow [2^n \rightarrow 2^n]$:

$$\begin{aligned} \text{Mac}(k, m_1 \cdots m_\ell) = & [(k_1, k_2) = \text{ext}(k); c_0 = 0; c_1 = \mathcal{F}(k_1)(c_0 \oplus m_1); \dots; \\ & c_\ell = \mathcal{F}(k_1)(c_{\ell-1} \oplus m_\ell) : \mathcal{F}(k_2)(c_\ell)] \end{aligned}$$

CPA-confidentialité + Intégrité : chiffrer et ensuite authentifier avec 2 clefs différentes.

TABLE 7.1 – Sommaire constructions en cryptographie symétrique

Chapitre 8

Systèmes asymétriques

En 1976, Diffie et Hellman [DH76] proposent une *nouvelle idée*. Chaque utilisateur dispose de deux clefs : une est gardée privée et l'autre est rendue publique. Si Alice veut envoyer un message à Bob elle le chiffre avec la clef publique de Bob. Seulement Bob pourra déchiffrer ce message en utilisant sa clef privée. Deux ans plus tard, Rivest, Shamir et Adleman [RSA78] proposent une première *réalisation concrète* de cette idée. Depuis, nombreux systèmes asymétriques ont été proposés. On se limite à présenter le système RSA mentionné ci dessus et le système d'ElGamal [Gam85]. Comme pour les système symétriques, une application naïve des systèmes asymétriques produit des systèmes cryptographiques faibles et inefficaces.

8.1 CPA confidentialité : cas asymétrique

La notion de *confidentialité-CPA* pour les systèmes asymétriques est similaire à celle donnée pour le cas symétrique (section 5.2). Les *différences principales* sont :

- Le générateur de clef G produit un *couple* (pk, sk) et les bits de pk dépendent des bits de sk .
- L'attaquant PPT *connaît* la clef pk , ce qui lui donne aussi accès à la *fonction de chiffrement* E_{pk} .¹

On suppose disposer d'un système cryptographique asymétrique qui assure la CPA-confidentialité de messages composés d'un bloc. Comment traite-t-on le chiffrement de messages composés de plusieurs blocs ? Un schéma simple mais inefficace consiste à appliquer le chiffrement bloc par bloc. Ainsi on a :

$$E_{pk}^{asym}(m_1 \cdots m_\ell) = E_{pk}^{asym}(m_1) \cdots E_{pk}^{asym}(m_\ell) .$$

Une solution beaucoup plus intéressante consiste à appliquer un schéma hybride qui combine le système asymétrique avec un système symétrique (qui est normalement plus efficace) :

$$E_{pk}^{hyb}(m_1 \cdots m_\ell) = [k' \leftarrow 2^n : (E_{pk}^{asym}(k'), E_{k'}^{sym}(m_1 \cdots m_\ell))] .$$

Dans le système hybride plus le message à chiffrer est long plus le coût du chiffrement s'approche du coût du système symétrique. On remarque aussi qu'à chaque chiffrement on génère

1. Dans les systèmes asymétriques l'attaquant connaît la fonction de chiffrement. Ainsi on arrive directement à la notion de CPA-confidentialité sans passer par la notion de 1-confidentialité.

une nouvelle clef symétrique et on s'attend à que la probabilité de générer deux fois la même clef est faible. On peut exploiter cette remarque pour montrer qu'il suffit de disposer d'un système symétrique qui assure la 1-confidentialité et d'un système asymétrique qui assure la CPA-confidentialité pour que le système hybride assure aussi la CPA-confidentialité. Ceci implique en particulier qu'on peut utiliser pour le système symétrique un système basé sur un *PRG*. Pour des longs messages, on approche le *coût* d'un chiffrement symétrique.

8.2 Système RSA

Dans le système RSA, on commence par générer deux grands *nombre premiers* p, q . Par exemple, on peut utiliser le test de Miller-Rabin (section A.8). Soit maintenant $n = pq$ le *module*. On choisit e *exposant de chiffrement* tel que $1 < e < \phi(n)$ et $\text{pgcd}(e, \phi(n)) = 1$. Ensuite, on calcule d *exposant de déchiffrement* tel que $(ed \equiv 1) \bmod \phi(n)$. L'algorithme d'Euclide nous donne une façon efficace d'effectuer ce calcul. Alors on a :

$$\text{Clef publique} = (n, e) , \quad \text{Clef privée} = (n, d) .$$

Comme textes *clairs* et textes *chiffrés* on prend :

$$\mathcal{P} = \mathcal{C} = \{m \mid 0 \leq m < n\} = \mathbf{Z}_n .$$

La fonction de *chiffrement* est définie par :

$$E_{(n,e)}(m) = m^e \bmod n ,$$

et pour la calculer on utilise l'algorithme du carré itéré (section A.5). De façon symétrique, la fonction de *déchiffrement* est définie par :

$$D_{(d,n)}(c) = c^d \bmod n ,$$

et à nouveau on utilise l'algorithme du carré itéré.

Il reste à vérifier que la fonction de déchiffrement est bien l'inverse à gauche de la fonction de chiffrement.

Proposition 57 *Soient n, e, d comme dans le système RSA. Alors pour tout $m \in \mathbf{Z}_n$ on a :*

$$(m^e)^d \equiv m \bmod n .$$

PREUVE. On sait :

$$(ed \equiv 1) \bmod \phi(n) .$$

Ce qui implique :

$$ed = 1 + l(p-1)(q-1) .$$

Ainsi :

$$(m^e)^d = m^{ed} = m^{1+l(p-1)(q-1)} = m(m^{(p-1)(q-1)})^l .$$

On montre que :

$$(m^{ed} \equiv m(m^{(p-1)})^{(q-1)l} \equiv m) \bmod p .$$

On génère deux grands nombres premiers (*différents !*) p, q . Soit $n = pq$ le *module*.

On choisit e *exposant de chiffrement* tel que $1 < e < \phi(n)$ et $\text{pgcd}(e, \phi(n)) = 1$.

L'*exposant de déchiffrement* d est l'inverse de e modulo $\phi(n)$.

$$\begin{aligned} \text{Clef Publique} &= (n, e) & \text{Clef Privée} &= (n, d) \\ E_{(n,e)}(m) &= m^e \bmod n & D_{(d,n)}(c) &= c^d \bmod n \quad \text{pour } m, c \in \mathbf{Z}_n \end{aligned}$$

TABLE 8.1 – Système RSA (version naïve)

On considère *deux cas*. Si $p \nmid m$ alors on applique le *petit théorème de Fermat*. D'autre part, si $p \mid m$ alors la congruence ci-dessus est *valide* car m est un multiple de p . De la même façon on montre :

$$(m^{ed} \equiv m(m^{(q-1)})^{(p-1)d} \equiv m) \bmod q .$$

Ainsi :

$$(m^{ed} \equiv m) \bmod p \quad (m^{ed} \equiv m) \bmod q .$$

Mais en général, pour p, q premiers différents on a :

$$(x_1 \equiv x_2) \bmod p \text{ et } (x_1 \equiv x_2) \bmod q \quad \text{implique} \quad (x_1 \equiv x_2) \bmod pq .$$

C'est-à-dire :

$$p \mid (x_1 - x_2) \text{ et } q \mid (x_1 - x_2) \quad \text{implique} \quad pq \mid (x_1 - x_2) .$$

□

La table 8.1 résume les ingrédients du système RSA.

Exemple 58 On choisit comme premiers $p = 11$ et $q = 23$. Alors le module est $n = pq = 253$ et

$$\phi(n) = (p-1)(q-1) = 4 \cdot 5 \cdot 11 .$$

Le plus petit exposant de chiffrement possible est $e = 3$ (en général, on ne peut pas avoir $e = 2$ car $\phi(n)$ est pair !). L'algorithme d'Euclide donne $d = 147$. L'espace des textes clairs et des textes chiffrés est $\{0, 1, \dots, 252\}$. Pour chiffrer le texte clair $m = 165$, on calcule $165^3 \bmod 253 = 110$. Pour déchiffrer le texte chiffré 110, on calcule $110^{147} \bmod 253$.

RSA avec bourrage aléatoire

Le système RSA décrit *ne satisfait pas* la propriété de *confidentialité CPA*. Par exemple, la fonction de chiffrement est *déterministe* ! Les systèmes RSA utilisés en pratique (par exemple, PKCS 1.5 ou OAEP) introduisent dans le chiffrement une *composante aléatoire* r qui est concaténée au message à chiffrer m . A titre indicatif, la taille de r est comparable à celle de m .

RSA et factorisation

L'attaquant connaît (n, e) . S'il peut factoriser $n = pq$, alors :

1. Il calcule $\phi(n) = (p-1)(q-1)$.
2. Il calcule d tel que $(ed \equiv 1) \bmod \phi(n)$ et il peut déchiffrer chaque message.

Ainsi RSA est sûr à condition qu'on ne puisse pas factoriser de façon efficace le produit de deux grands nombres premiers. Par ailleurs, on peut montrer que si l'on connaît d alors il y a un algorithme probabiliste pour factoriser n . Ainsi apprendre d est aussi difficile que factoriser n . Cependant, il pourrait y avoir d'autres méthodes pour obtenir de l'information sur le texte clair qui ne nécessitent pas le calcul de d . A l'état de nos connaissances, attaquer RSA pourrait être *plus facile* que factoriser le produit de deux grands nombres premiers.

8.3 Échange de clefs : protocole de Diffie-Hellman

On présente un *protocole* dû à Diffie et Hellman qui permet à deux participants qui partagent seulement de l'*information publique* d'établir une *clef commune secrète*. On verra ensuite comment dériver du protocole un système asymétrique.

1. Alice et Bob se mettent d'accord sur un *grand nombre premier* p et un *générateur* g pour \mathbf{Z}_p^* .
2. Alice choisit de façon aléatoire $a \in \mathbf{Z}_p^*$, et elle calcule et envoie à Bob $A = (g^a) \bmod p$.
3. Bob choisit de façon aléatoire $b \in \mathbf{Z}_p^*$, il calcule et envoie à Alice $B = (g^b) \bmod p$.
4. La *clef commune* $K = g^{ab} \bmod p$ est extraite :

$$K = B^a \bmod p \quad (\text{comme Alice calcule } K)$$

$$K = A^b \bmod p \quad (\text{comme Bob calcule } K)$$

Exercice 59 On suppose que $p = 17$, $g = 3$, Alice choisit $a = 7$, et Bob $b = 4$. Quelle est la clef commune générée par le protocole ?

Le problème de Diffie-Hellman

Le problème de Diffie-Hellman est le suivant :

étant donné (p, g, A, B) (mais en ignorant le logarithme discret de A et B) peut-on déterminer la clef K ?

Clairement, si on peut calculer le *logarithme discret* de façon efficace alors on peut déterminer K (et résoudre le problème de Diffie-Hellman). Par exemple, on calcule $a = d\log_g(A)$ et $K = (B^a) \bmod p$. Par contre, on *ne sait pas* si la solution du problème de DH est suffisante pour calculer le logarithme discret de façon efficace. La situation est donc similaire à celle décrite pour RSA : à défaut de savoir prouver quelque chose, on se réduit à un problème un peu plus facile.

Attaque de l'homme au milieu (man in the middle)

Le protocole de Diffie-Hellman est conçu pour résister à un attaquant *passif*. Que se passe-t-il si l'attaquant est *actif*, c'est-à-dire s'il peut intercepter et générer des messages ?

- L'attaquant exécute le protocole d'échange de clef avec Alice en jouant le *rôle de Bob* et avec Bob en jouant le *rôle d'Alice*.
- A la fin des deux protocoles, Alice et Bob vont envoyer des messages chiffrés avec deux *clefs connues par l'attaquant*.
- A remarquer que cette attaque n'est pas basée sur une manipulation de l'information publique (g, p) . Le problème est l'*authentification* du correspondant.

8.4 Système d'ElGamal

Il s'agit d'un système à *clef publique* basé sur le problème de Diffie-Hellman et effectivement utilisé (par exemple dans la bibliothèque GPG).

Génération de la Clef

1. Alice génère un *grand nombre premier* p et un générateur g pour \mathbf{Z}_p^* .
2. Ensuite elle choisit de façon aléatoire un *exposant* $a \in \mathbf{Z}_p^*$ et elle calcule :

$$A = g^a \bmod p .$$

La *clef publique* est (p, g, A) . La *clef secrète* est l'exposant a .

Chiffrement Bob connaît (p, g, A) . Pour chiffrer un message $m \in \mathbf{Z}_p$ pour Alice, il choisit un *exposant aléatoire* $b \in \mathbf{Z}_p^*$ et il calcule (jusqu'ici c'est comme l'échange de clef!) :

$$B = (g^b) \bmod p .$$

Pour *chiffrer* m , Bob calcule (remarquez que A^b était la clef dans DH) :

$$c = A^b m \bmod p .$$

Ainsi le code chiffré comprend le produit de la clef DH avec m modulo p :

$$E_{(p,g,A)}(m) = (B, c) .$$

Déchiffrement Alice obtient (B, c) et elle connaît a . Pour calculer m , elle divise c par $B^a \bmod p$ comme suit :

1. Calculer (une fois) $x = p - 1 - a$, $1 \leq x \leq p - 1$.
2. Calculer $B^x c \bmod p$ (ceci est égal à m).

On vérifie que $B^x c \bmod p = m$ pour $x = p - 1 - a$:

$$\begin{aligned} B^x c &\equiv g^{b(p-1-a)} A^b m \\ &\equiv (g^{p-1})^b (g^{-ab}) (g^{ab}) m \\ &\equiv (1)^b (1) m \\ &\equiv m \bmod p . \end{aligned}$$

On remarquera que le chiffrement dans ce système est un algorithme probabiliste. Comme dans le cas symétrique, cela est nécessaire pour assurer la CPA-confidentialité.

Fait 60 Si le problème de Diffie-Hellman est difficile alors le chiffrement d'El-Gamal assure la CPA confidentialité.

La table 8.2 résume le système cryptographique d'El-Gamal.

Générer :

$$p \text{ premier} \quad g \text{ générateur pour } \mathbf{Z}_p^* \quad a \in \mathbf{Z}_p^*$$

Calculer :

$$\text{Clef Publique : } (p, g, A = (g^a) \bmod p) \quad \text{Clef Privée : } a$$

Chiffrement et Déchiffrement :

$$\begin{aligned} \text{Texte clair } \mathcal{P} = \mathbf{Z}_p \quad \text{Texte chiffré } \mathcal{C} = \mathbf{Z}_p \times \mathbf{Z}_p \\ E_{(p,g,A)}(m) = [b \leftarrow \mathbf{Z}_{p-1} : (g^b \bmod p, (A^b \cdot m) \bmod p)] \\ D_{(p,g,a)}(B, c) = (B^{p-1-a} \cdot c) \bmod p \end{aligned}$$

TABLE 8.2 – Système d'ElGamal

8.5 Sommaire

Dans un système de chiffrement asymétrique, on distingue deux clefs dont l'une, publique, sert à chiffrer et l'autre, privée, sert à déchiffrer. Dans ces systèmes il doit être difficile de calculer la clef privée à partir de la clef publique. On a présenté deux systèmes qui mettent en oeuvre cette idée : le système RSA qui est lié à la difficulté de factoriser le produit de deux grands nombres premiers et le système d'ElGamal qui est lié à la difficulté à calculer l'inverse de l'exposant modulaire.² En pratique, on peut combiner les systèmes asymétriques avec les systèmes symétriques selon un schéma hybride qui offre les avantages des systèmes asymétriques avec un coût proche des systèmes symétriques.

2. Bien sur, être lié n'est pas la même chose que être équivalent !

Chapitre 9

Hachage et compression

Soit Σ un alphabet donné. Une *fonction de hachage* est une fonction :

$$h : \Sigma^* \rightarrow \Sigma^m \quad (\text{fonction de hachage}) \quad (9.1)$$

Une *fonction de compression* est une fonction :

$$h : \Sigma^{t+m} \rightarrow \Sigma^m \quad t \geq 1 \quad (\text{fonction de compression}) \quad (9.2)$$

Les fonctions de hachage sont utilisées dans plusieurs contextes. En cryptographie, une fonction de hachage doit avoir certaines propriétés particulières (voir le protocole pour jouer à pile ou face au téléphone, section 1.6) :

1. Pour un x donné, il devrait être facile de calculer $h(x)$.
2. Pour un y donné, il devrait être difficile de trouver une *image inverse* x telle que $h(x) = y$, c'est-à-dire h est à *sens unique* (voir définition 38 pour une formalisation). Par exemple, l'exponentiation modulaire est facile à calculer mais le logarithme discret est difficile (autant qu'on sache...).
3. Pour un x donné, il devrait être difficile de trouver un autre x' tel que $h(x) = h(x')$. Si c'est le cas, on dit que h est *faiblement résistante aux collisions*.
4. Il devrait être difficile de trouver x, x' tels que $x \neq x'$ et $h(x) = h(x')$. Dans ce cas on dit que h est *fortement résistante aux collisions*.

Un exemple dans lequel la résistance faible aux collisions suffit est le suivant : on veut protéger un programme important de toute modification (*e.g.*, le programme qui calcule le chiffrement). Dans ce cas, on peut mémoriser le hachage du programme dans un endroit sûr (par exemple, une *smart card*). Avant d'utiliser le programme, on compare toujours le hachage du programme à la valeur mémorisée dans un endroit sûr. Ceci suppose que h est faiblement résistante aux collisions. On verra dans la suite du cours que la propriété de résistance forte aux collisions trouve des applications dans la conception de signatures digitales (chapitre 10).

On peut montrer que les fonctions fortement résistantes aux collisions sont à sens unique (définition 38). Voici l'argument informel :

1. On suppose disposer d'un algorithme d'inversion.
2. On choisit x' de façon aléatoire. En utilisant l'algorithme d'inversion, on détermine un x tel que $h(x) = h(x')$.
3. (x, x') va être une collision sauf si $x = x'$, ce qui est peu probable.

9.1 Paradoxe des anniversaires

Pour calibrer une fonction de hachage il est utile de rappeler un fait élémentaire de calcul des probabilités connu comme *paradoxe des anniversaires*.

Combien de personnes faut-il réunir pour que la probabilité de trouver deux personnes avec la même date d'anniversaire soit supérieure ou égale à $1/2$? Soient n le nombre de *jours* dans une année et k le nombre de *personnes*. Une *expérience* est un élément de $\{1, \dots, n\}^k$. Toutes les expériences ayant la même probabilité, le calcul des probabilités se réduit à un problème de comptage. A savoir il faut *compter* le nombre d'expériences (d_1, \dots, d_k) telles que $d_i \neq d_j$ si $i \neq j$. Ce nombre est :

$$n(n-1) \cdots (n-k+1) = \prod_{i=0, \dots, k-1} (n-i) .$$

Donc la probabilité q qu'il *n'y ait pas deux personnes* avec la même date d'anniversaire est :

$$q = \left(\frac{1}{n^k}\right) \prod_{i=0, \dots, k-1} (n-i) = \prod_{i=0, \dots, k-1} \frac{(n-i)}{n} = \prod_{i=1, \dots, k-1} \left(1 - \frac{i}{n}\right) .$$

On *borne supérieurement* $1+x$ par e^x et on dérive :

$$q \leq \prod_{i=1, \dots, k-1} e^{-\frac{i}{n}} = e^{-\sum_{i=1, \dots, k-1} \left(\frac{i}{n}\right)} = e^{-\frac{k(k-1)}{2n}} .$$

Pour avoir $q \leq 1/2$, il suffit que :

$$\frac{-k(k-1)}{2n} \leq -\ln(2) .$$

On étudie l'*inégalité quadratique* :

$$k^2 - k - 2n \ln(2) \geq 0 .$$

On dérive que pour avoir $q \leq 1/2$, il suffit que :

$$k \geq \frac{1 + \sqrt{(1 + 8n \ln(2))}}{2} .$$

En particulier, pour $n = 365$, il *suffit* d'avoir $k = 23$ (en général il suffit de prendre $k \approx (1, 2)\sqrt{n}$).

On considère maintenant les implications pour les fonctions de hachage. Si $\Sigma = \{0, 1\}$, alors le nombre de *valeurs de hachage* (les dates d'anniversaire) est 2^n et l'*inégalité* devient :

$$k \geq \frac{1 + \sqrt{1 + 8 \ln(2) 2^n}}{2} .$$

Ceci signifie que si on calcule un peu moins que $2^{n/2}$ *valeurs* alors la probabilité de trouver une collision est *plus grande* que $1/2$. Actuellement, on *recommande* $n \geq 256$.

9.2 Construction de Merkle-Damgård

On dit qu'une fonction de hachage (ou de compression) est *résistante aux collisions* s'il n'y a pas d'algorithme probabiliste polynomial en temps qui trouve une collision.¹ On ne sait pas si de telles fonctions existent. Cependant, une construction due à Merkle et Damgård [Mer89, Dam89] montre que si les *fonctions de compression existent* alors les *fonctions de hachage existent* aussi.

1. Pour formaliser cette intuition, il faudrait considérer des *familles* de fonctions de hachage.

Proposition 61 Soit $\Sigma = \{0, 1\}$ un alphabet binaire. Soit $c : 2^{t+m} \rightarrow 2^m$ est une fonction de compression résistante aux collisions avec $t \geq 1$. Alors on peut construire une fonction de hachage résistante aux collisions :

$$h : \bigcup_{i > t+m} 2^i \rightarrow 2^m .$$

En supposant $n = |x|$, le nombre d'appels à c pour calculer $h(x)$ est au plus :

$$\begin{array}{ll} 1 + \lceil n/(t-1) \rceil & \text{si } t \geq 2 \\ 2n + 2 & \text{si } t = 1 . \end{array}$$

Remarque 62 En pratique $t \geq 2$ et donc on applique la compression environ n/t fois.

On présente une construction simplifiée pour $t \geq 64$. Ici on choisit 64 car l'on suppose que 64 bits suffisent pour représenter le nombre de blocs dans le message à hacher.

- Soit $c : 2^{t+m} \rightarrow 2^m$ la fonction de compression.
- Soit $x = x_1 \cdots x_n$ la suite de bits à hacher (avec $n = k \cdot t$, x est donc composé de k blocs de t bits).
- On fait le bourrage (*padding*) suivant de x

$$y(x) = x_1 \cdots x_n \cdot 0 \cdots 0 \cdot (k)_2$$

où $0 \cdots 0(k)_2$ est la représentation en base 2 de k (le nombre de blocs de x) avec assez de 0 à gauche pour avoir 64 bits.

- Ainsi :

$$y(x) = B_1 \cdots B_{k+1} \text{ où } B_i \in 2^t .$$

- Soit $H_0 = IV \in 2^m$ un vecteur d'initialisation de m bits.
- Alors on calcule $h(x) = H_{k+1}$ où :

$$H_{i+1} = c(B_{i+1}, H_i) \quad (i = 0, \dots, k) .$$

On montre que si on sait trouver une collision de h alors on peut trouver dans un temps raisonnable une collision de c .

- Soient x et x' deux textes (pas forcément de la même longueur) tels que $h(x) = h(x')$.
- On suppose que x est constitué de k blocs et x' de k' blocs.
- Par hypothèse on a $h(x) = H_{k+1} = H'_{k'+1} = h(x')$. Si $k \neq k'$ alors $c((k)_2, H_k) = c((k')_2, H_{k'})$ est une *collision*.
- Sinon on doit avoir $k = k'$. Soit i le plus grand indice tel que $B_i \neq B'_i$.
- Si $H_k \neq H'_k$ on a une *collision*.
- Sinon $c(B_k, H_{k-1}) = H_k = H'_k = c(B'_k, H'_{k-1})$ et soit $B_k \neq B'_k$ (collision) soit $H_{k-1} \neq H'_{k-1}$ (collision) soit $H_{k-1} = H'_{k-1}$.

En itérant, on voit qu'on arrive forcément au i tel que $B_i \neq B'_i$ et on trouve ainsi une *collision*.

9.3 Compression de Davies-Meyer

On ne sait pas si les fonctions de compression résistantes aux collisions existent. En pratique, on se contente de construire une fonction de compression à partir d'une fonction de *chiffrement*. Par exemple, on suppose que :

$$E : 2^t \times 2^m \rightarrow 2^m ,$$

est un chiffrement à bloc avec clef de taille t , bloc de taille m et fonction inverse $D : 2^t \times 2^m \rightarrow 2^m$. *Rappel* avec AES on a : taille de la clef $t \in \{128, 192, 256\}$ et taille du bloc $m = 128$.

En suivant *Davies-Meyer* [MMO85], on dérive une fonction de compression c :

$$c : 2^{t+m} \rightarrow 2^m \quad c(x, y) = E_x(y) \oplus y .$$

Si on applique la construction de Merkle-Damgård avec cette fonction de compression on obtient :

$$H_0 = IV \quad H_{i+1} = c(B_i, H_i) = E_{B_i}(H_i) \oplus H_i .$$

A noter qu'on utilise le bloc B_i du texte à hacher comme clef.

On peut montrer que si les fonctions de chiffrement sont *distribuées de façon uniforme* (une hypothèse forte !) alors la recherche d'une collision demande un *temps* $O(2^{m/2})$ (qui n'est pas mieux que le temps nécessaire à une attaque par force brute !).

D'autres *schémas* similaires existent (Matyas et al., Miyagushi et al...) mais pas tous les schémas marchent... Par exemple, si l'on pose :

$$c'(x, y) = E_x(y) \quad (\text{mauvaise compression !}) ,$$

il est *facile de trouver une collision* en prenant x, y, x' arbitraires et $y' = D_{x'}(E_x(y))$ car :

$$c'(x, y) = E_x(y) = E_{x'}(D_{x'}(E_x(y))) = c'(x', y') .$$

Voici des données sur la taille du hash et la vitesse relative de différentes fonctions de hachage.

| Fonction de hachage | longueur du bloc | vitesse relative |
|---------------------|------------------|------------------|
| MD4 | 128 | 1.00 |
| MD5 | 128 | 0.68 |
| SHA-1 | 160 | 0.28 |
| SHA-2 | 256, ... | 0.14 |

Une collision pour *MD4* peut être trouvée en calculant 2^{20} valeurs de hachage. Une collision de la fonction de compression de *MD5* a été trouvée aussi. Plus récemment (2004), une équipe chinoise a proposé une méthode pour trouver une collision dans *SHA-1* en 2^{69} essais. Ceci est 2000 fois plus rapide que les 2^{80} essais demandés par une méthode de recherche complète. Par ailleurs, d'autres avancées ont forcé le passage de *SHA-1* à *SHA-2*. De plus, en 2012 un standard alternatif nommé *SHA-3* a été sélectionné par un processus de compétition ouverte similaire à celui utilisé pour AES.

9.4 Compression CHP

Pour construire une fonction de compression on peut aussi s'inspirer des méthodes de chiffrement développées pour les systèmes asymétriques. En particulier, pour une fonction

de compression proposée par Chaum, van Heijst et Pfitzmann (abrégé en compression CHP) [CvHP91] on peut montrer qu'elle résiste aux collisions si le calcul du *logarithme discret* dans $(\mathbf{Z}_p)^*$ est difficile.

Définition 63 (compression CHP) Soient p et $q = (p-1)/2$ premiers et a un générateur pour $(\mathbf{Z}_p)^*$. Soit b choisi de façon aléatoire dans \mathbf{Z}_p^* . Alors la compression CHP est définie par :

$$c : \mathbf{Z}_q \times \mathbf{Z}_q \rightarrow \mathbf{Z}_p \quad c(x_1, x_2) = a^{x_1} b^{x_2} \mod p .$$

Remarque 64 Comme $q = (p-1)/2$, la taille de (x_1, x_2) est approximativement deux fois la taille de p . Malheureusement cette fonction est basée sur une opération assez lourde (l'exponentiation modulaire) et elle ne semble pas assez efficace pour être utilisée en pratique.

Exercice 65 On suppose $q = 11$, $p = 23$, $a = 5$, et $b = 4$. Calculez $c(5, 10)$.

Pour réduire le calcul du logarithme discret à la recherche d'une collision on utilise une généralisation du théorème de Bezout (proposition 77) dont la preuve est donnée en appendice qui permet de dériver le corollaire suivant.

Corollaire 66 Si $2 = \text{pgcd}(a, m)$ et $2 \mid b$ alors la congruence $(ax \equiv b) \mod m$ a exactement 2 solutions modulo m .

PREUVE. Si x' est la solution de :

$$(a/2)x' \equiv (b/2) \mod (m/2) ,$$

alors les solutions de la congruence de départ ont la forme :

$$x' + k(m/2) ,$$

et il y en a exactement 2 dans \mathbf{Z}_m . □

Proposition 67 On peut dériver d'un algorithme efficace pour la recherche d'une collision de la compression CHP un algorithme efficace pour le calcul du logarithme discret.

PREUVE. Une *collision* est un couple (x_1, x_2) et (x_3, x_4) tel que :

$$(a^{x_1} b^{x_2} \equiv a^{x_3} b^{x_4}) \mod p .$$

Ceci implique :

$$(a^{x_1-x_3} \equiv b^{x_4-x_2}) \mod p .$$

On montre comment à partir d'une telle collision on peut calculer le *logarithme discret* :

$$y = d\log_a(b) \text{ modulo } p .$$

Comme $b = a^y$, la congruence précédente donne :

$$(a^{x_1-x_3} \equiv a^{y(x_4-x_2)}) \mod p .$$

Comme a est un *générateur* pour $(\mathbf{Z}_p)^*$, ceci implique :

$$(x_1 - x_3 \equiv y(x_4 - x_2)) \pmod{p-1} \text{ et } (p-1) = 2q .$$

La congruence a une solution si et seulement si $g = \text{pgcd}(x_4 - x_2, p-1)$ divise $x_1 - x_3$ (par le corollaire 66). On sait que $|x_4 - x_2| < q$ car $x_2, x_4 \in \mathbf{Z}_q$. Comme $p-1 = 2q$ et q est premier, on dérive que :

$$\begin{cases} g = 1 & \text{si } |x_4 - x_2| \text{ est impair} \\ g = 2 & \text{autrement.} \end{cases}$$

Si $g = 1$ alors la solution modulo $p-1$ de la congruence précédente est unique et peut être calculée. Ceci permet de déterminer le logarithme discret. Si $g = 2$ alors la congruence a deux solutions différentes (voir corollaire 66) modulo $p-1$ et le logarithme discret peut être calculé en essayant les deux. \square

Remarque 68 *Les utilisateurs de la fonction de compression CHP doivent se mettre d'accord sur les paramètres a et b . Si un de ces utilisateurs peut choisir b ou s'il peut calculer le logarithme discret de b alors il peut trouver des collisions (exercice!).*

9.5 Sommaire

L'espace d'hachage doit être suffisamment large pour résister à l'*attaque des anniversaires*. En général, il s'agit d'un espace plus grand que pour le chiffrement symétrique. Si vous pouvez *compresser* alors vous pouvez *hacher* en sécurité grâce à la construction de Merkle-Damgård. Il est possible de construire des fonctions de compression sûres dans un sens théorique. Par exemple, pour la compression CHP on peut montrer que la recherche d'une collision est aussi difficile que le calcul du logarithme discret. Cependant de telles fonctions sont considérées comme trop coûteuses. En pratique, on peut dériver des fonctions de compression des systèmes de *chiffrement symétriques*. Le construction de Davies-Meyer est une façon de procéder. Il y a du progrès du côté des attaquants ce qui entraîne la définition de nouveaux standards : SHA-2, SHA-3, ... et l'élargissement de l'espace de hachage (256, 512, ... bits)

Chapitre 10

Intégrité : cas asymétrique

On commence par rappeler les idées générales.

- Alice a une *clef publique* e et une *clef privée* d . Pour *signer* un document m elle calcule $s(d, m)$.
- Bob reçoit le couple $(m, s(d, m))$. En utilisant la *clef publique* e , il peut vérifier que le message a vraiment été signé par Alice.

Comme pour les MAC (cas symétrique), Bob ayant vu un nombre polynomial de messages avec leur signature doit être incapable de produire une signature valide d'Alice pour un nouveau message.

Remarque 69 *Dans ce schéma : la clef privée est utilisée quand on envoie le message, et la clef publique (de l'émetteur) quand on le reçoit. C'est juste le contraire de ce qui se passe dans les systèmes de chiffrement asymétriques !*

10.1 Signature RSA

On décrit un schéma de signature basé sur le système RSA. On remarquera que dans RSA les clefs de 'chiffrement' et de 'déchiffrement' ont les *mêmes* propriétés (chaque clef est l'inverse de l'autre). Les noms se réfèrent à *l'utilisation envisagée* et non pas à des propriétés mathématiques particulières.

Voici une première version (faible!) d'un système de signature RSA (voir table 8.1). Les clefs publique et privée d'Alice sont déterminées comme dans RSA. Alice signe le document m en calculant :

$$s = s(d, m) = (m^d) \bmod n .$$

Bob reçoit (m, s) et vérifie si :

$$(s^e \equiv m) \bmod n .$$

Un attaquant, devrait être capable de calculer la racine e modulo n du document m . On ne connaît pas d'algorithme efficace pour ce problème.

Exercice 70 *Montrez que si l'on connaît la signature RSA faible de m_1 et m_2 alors on sait calculer la signature RSA faible de $(m_1 \cdot m_2) \bmod n$. Donc RSA faible ne garantit pas l'intégrité !*

Remarque 71 *Comme dans les systèmes de chiffrement à clef publique, il est important que Bob obtienne réellement la clef publique d'Alice. Si, à sa place, Bob reçoit une clef publique produite par un attaquant, alors l'attaquant peut produire des messages arbitraires et faire croire à Bob qu'ils ont été signés par Alice.*

On va maintenant introduire un système de signature RSA plus pratique et plus robuste qui utilise une fonction de hachage. En effet, pour l'instant on a supposé que le message m est un élément de \mathbf{Z}_n . Que fait-on si m est un message de longueur arbitraire ? L'approche est de hacher le message avec une fonction fortement résistante aux collisions (chapitre 9) et ensuite de signer le hachage.

- La signature d'Alice du message m est :

$$s = (h(m)^d) \bmod n .$$

On remarque que maintenant le message m ne peut pas être dérivé de la signature.

- Bob reçoit (x, s) et vérifie :

$$(s^e) \bmod n = h(x) .$$

Que fait-on si maintenant Alice veut envoyer un message confidentiel et authentifié à Bob ? Une possibilité est de suivre la même approche évoquée dans le cas symétrique (section 6.3) : à savoir d'abord on chiffre et ensuite on signe. Ce qui veut dire que Alice chiffre le message avec la clef publique (chiffrement) de Bob et ensuite elle signe le résultat avec sa clef privée (signature). Bob vérifie la signature avec la clef publique (signature) d'Alice et ensuite déchiffre avec sa clef privée (chiffrement).

Remarque 72 *On suppose que la fonction de hachage résiste aux collisions.*

- Trouver un message x qui correspond à une signature s , revient à trouver x tel que $h(x) = (s^e) \bmod n$, c'est-à-dire, à déterminer l'image inverse de (s^e) .
- Dans le cas de plagiat qui exploite la propriété multiplicative de RSA, il est improbable que $h(m_1 m_2) = h(m_1) h(m_2)$.
- Enfin, on note que remplacer (x, s) par (x', s) revient à trouver x' tel que $h(x) = h(x')$.

10.2 Sommaire

Les *signatures* sont les systèmes qui garantissent l'intégrité des messages dans le cas asymétrique. Ces systèmes combinent les chiffrements asymétriques avec les fonctions de hachage. On a considéré un système de signature particulièrement simple basé sur le système RSA. Un système de signature basé sur le système d'ElGamal existe aussi et une version optimisée (*DSA* pour *Digital signature algorithm*) est disponible dans le logiciel *PGP*.

Sommaire : cryptographie asymétrique

La table suivante résume les constructions présentées dans le cas asymétrique.

Chiffrement hybride :

$$E_{pk}^{hyb}(m_1 \cdots m_\ell) = [k' \leftarrow 2^n : (E_{pk}^{asym}(k'), E_{k'}^{sym}(m_1 \cdots m_\ell))] .$$

Chiffrement RSA : $n = pq$, $\text{pgcd}(e, \phi(n)) = 1$, $(de \equiv 1) \bmod n$, m = message + composante aléatoire (standard *OAEP*) :

$$E_{(e,n)}(m) = (m^e) \bmod n .$$

Échange de clef : Diffie-Hellman. g générateur pour \mathbf{Z}_p^* . $K = (g^{ab}) \bmod p$, où :

$$\begin{aligned} Alice \rightarrow Bob : & [a \leftarrow \mathbf{Z}_p^* : (g^a) \bmod p] \\ Bob \rightarrow Alice : & [b \leftarrow \mathbf{Z}_p^* : (g^b) \bmod p] \end{aligned}$$

Chiffrement El Gamal. g générateur pour \mathbf{Z}_p^* :

$$E_{p,g,A}(m) = [b \leftarrow \mathbf{Z}_p^* : ((g^b) \bmod p, (A^b \cdot m) \bmod p)]$$

Compression Meyer-Davies en utilisant un chiffrement à bloc (AES) :

$$c(x, y) = E_x(y) \oplus y \quad c : 2^{t+m} \rightarrow 2^m$$

Hachage Merkle-Damgård. Pour hacher x , à partir de $c : 2^{t+m} \rightarrow 2^m$, on fait un bourrage avec insertion de la longueur de x pour obtenir : $y(x) = B_1 \cdots B_{k+1}$ où $B_i \in 2^t$. Ensuite, $h(x) = H_{k+1}$ où $H_0 = IV$ et :

$$H_{i+1} = c(B_{i+1}, H_i) \quad (i = 0, \dots, k)$$

Signature RSA avec h fonction de hachage :

$$\text{Sig}_{(d,n)}(m) = (h(m)^d) \bmod n$$

Annexe A

Arithmétique

Dans ce chapitre on collecte des résultats élémentaires d'arithmétique modulaire qui sont nécessaires à la compréhension des constructions cryptographiques présentées dans les notes de cours.

Convention On dénote par \mathbf{Z} l'ensemble des nombres entiers et si $m \in \mathbf{Z}$ et $m \geq 1$ on dénote par \mathbf{Z}_m l'ensemble $\{0, 1, \dots, m-1\}$ des entiers modulo m .

A.1 Division et reste

Définition 73 (divisibilité) Soient $a, b \in \mathbf{Z}$. On dit que a divise b et on écrit $a \mid b$ si $\exists c \in \mathbf{Z} \ ca = b$. De façon équivalente, on dit aussi que b est un multiple de a .

Proposition 74 Soient $a, b, c \in \mathbf{Z}$. Alors :

1. $a \mid 0$.
2. Si $0 \mid a$ alors $a = 0$.
3. Si $a \mid b$ et $b \mid c$ alors $a \mid c$.
4. Si $c \mid a$ et $c \mid b$ alors pour tout $d, e \in \mathbf{Z}$ ($c \mid da + eb$).
5. Si $a \mid b$ et $b \neq 0$ alors $|a| \leq |b|$.
6. Si $a \mid b$ et $b \mid a$ alors $|a| = |b|$.

PREUVE. (1) On prend $c = 0$. (2) Comme $0c = 0 = a$. (3) $k_1a = b$ et $k_2b = c$ implique $k_1k_2a = c$. (4) $k_1c = a$ et $k_2c = b$ implique $da + eb = dk_1c + ek_2c = (dk_1 + ek_2)c$. (5) $ca = b$ et $b \neq 0$ implique $c \neq 0$, $a = ca/c = b/c \leq b$. (6) Pour $a, b \neq 0$ on applique (5); et $a = 0$ ssi $b = 0$. \square

Notation Si $x \in \mathbf{R}$ alors $\lfloor x \rfloor = \max\{a \in \mathbf{Z} \mid a \leq x\}$. Notez que $\lfloor 3,5 \rfloor = 3$ et $\lfloor -3,5 \rfloor = -4$.

Proposition 75 Soient $a, b \in \mathbf{Z}$ avec $b > 0$. Alors :

$$\exists! q, r \ (a = qb + r \text{ et } 0 \leq r < b).$$

PREUVE. On montre d'abord l'existence. Soit $r = a - \lfloor a/b \rfloor b$. On vérifie : $a = \lfloor a/b \rfloor b + (a - \lfloor a/b \rfloor b)$ et $0 \leq r < b$. Pour l'unicité, on suppose $0 \leq r < b$ et $q \in \mathbf{Z}$ tel que $a = qb + r$. Alors $r = (a - qb)$ et $0 \leq (a - qb) < b$. Comme $b > 0$, on dérive $0 \leq a/b - q < 1$. C'est-à-dire $q \leq a/b < q + 1$ avec $q \in \mathbf{Z}$ et $a/b \in \mathbf{Q}$. Ceci implique que $\lfloor a/b \rfloor = q$, et q ne dépend pas de r . Donc si $a = q_1b + r_1 = q_2b + r_2$, avec $0 \leq r_1, r_2 < b$. Alors $q_1 = q_2 = \lfloor a/b \rfloor$. Il suit que : $r_1 = r_2$. \square

Notation Dans les hypothèses de la proposition 75, on écrit $a \bmod b$ pour le reste r de la division entière.

A.2 Plus grand commun diviseur (pgcd)

Si $c \mid a$ et $c \mid b$ on dit que c est un diviseur commun de a et b . Si $a \neq 0$ ou $b \neq 0$ alors on sait que les diviseurs de a et b sont bornés en valeur absolue par $\max(|a|, |b|)$. Si $a = b = 0$ alors le plus grand commun diviseur n'existe pas car $\forall c \ (c \mid 0)$, mais par convention on peut établir que $\text{pgcd}(0, 0) = 0$. On arrive donc à la définition suivante.

Définition 76 (pgcd) Soient $a, b \in \mathbf{Z}$. On définit le plus grand diviseur commun (pgcd) par :

$$\text{pgcd}(a, b) = \begin{cases} 0 & \text{si } a = b = 0 \\ \max\{c \mid (c \mid a) \text{ et } (c \mid b)\} & \text{autrement.} \end{cases}$$

Notation On pose :

$$k\mathbf{Z} = \{kc \mid c \in \mathbf{Z}\}, \quad X + Y = \{x + y \mid x \in X \text{ et } y \in Y\}.$$

Avec cette notation, $a\mathbf{Z} + b\mathbf{Z}$ est l'ensemble des nombres entiers obtenus comme combinaisons linéaires de a et b . Le théorème de Bezout affirme que les combinaisons linéaires de deux entiers sont exactement les multiples de leur pgcd.

Proposition 77 Soient $a, b \in \mathbf{Z}$. Alors :

$$a\mathbf{Z} + b\mathbf{Z} = \text{pgcd}(a, b)\mathbf{Z}.$$

PREUVE. Soient $A = a\mathbf{Z} + b\mathbf{Z}$ et $I = \text{pgcd}(a, b)\mathbf{Z}$. Si $a = b = 0$ alors $A = \{0\} = I$. Donc on suppose $a \neq 0$ ou $b \neq 0$ et on définit :

$$g = \min\{xa + yb \mid x, y \in \mathbf{Z}, xa + yb > 0\}.$$

Clairement, $g\mathbf{Z} \subseteq a\mathbf{Z} + b\mathbf{Z}$. D'autre part, soit $c = ak_1 + bk_2 \neq 0$. On sait que $\exists q, r \ c = qg + r, 0 \leq r < g$. Donc :

$$\begin{aligned} r &= ak_1 + bk_2 - q(ax + by) \\ &= a(k_1 - qx) + b(k_2 - qy) \in a\mathbf{Z} + b\mathbf{Z}. \end{aligned}$$

Comme g est le plus petit entier positif, on doit avoir $r = 0$. Donc $c = qg \in g\mathbf{Z}$ et $a\mathbf{Z} + b\mathbf{Z} \subseteq g\mathbf{Z}$. Ensuite on montre que $g = \text{pgcd}(a, b)$. Comme $a, b \in g\mathbf{Z}$, il suit que $g \mid a$ et $g \mid b$. D'autre part, si $d \mid a$ et $d \mid b$ alors $d \mid ax + by = g$. Comme $g \neq 0$, on sait que $|d| \leq g$, donc g est le plus grand commun diviseur. \square

Exercice 78 1. Montrez que l'équation $ax + by = c$ a une solution en x, y ssi $\text{pgcd}(a, b) \mid c$.

2. Pour tout $a, b \in \mathbf{Z}$, ils existent $x, y \in \mathbf{Z}$ tels que $\text{pgcd}(a, b) = ax + by$.

3. Soient $a, b \in \mathbf{Z}$ avec $\text{pgcd}(a, b) > 0$. Alors $d \mid a$ et $d \mid b$ implique que $d \mid \text{pgcd}(a, b)$.

Le résultat suivant permet d'élargir le champ d'application de la proposition 77 (Bezout).

Proposition 79 Soit :

$$(ax \equiv b) \pmod{m}, \tag{A.1}$$

une congruence et soit $g = \text{pgcd}(a, m)$. Alors on a exactement deux cas :

1. Si g ne divise pas b la congruence n'a pas de solution.

2. Sinon, soient $a' = a/g$, $b' = b/g$ et $m' = m/g$ et soit x' la solution unique modulo m' de la congruence :

$$(ax' \equiv b) \pmod{m'}.$$

Alors les solutions de la congruence (A.1) sont exactement celles de la forme : $x' + km'$.

PREUVE. La preuve se décompose dans les étapes suivantes.

1. Si $(ax \equiv b) \bmod m$ a une solution alors $g = \text{pgcd}(a, m)$ divise b .
En effet si $ax - b = km$ alors $ax + (-k)m = b$ et par Bezout $b = l \cdot g$.
2. Si $\text{pgcd}(a, m) = 1$ alors la congruence $(ax \equiv b) \bmod m$ a une solution unique modulo m .
Car a a une inverse modulo m et donc $(x \equiv a^{-1}b) \bmod m$.
3. Si $g \mid b$ alors la congruence dérivée $(a'x \equiv b') \bmod m'$ a une solution unique, disons x' .
Comme $\text{pgcd}(a', m') = 1$ on applique 2.
4. Si $g \mid b$ alors $x' + ym'$ est une solution de $ax \equiv b \bmod m$ pour tout $y \in \mathbf{Z}$.
On remarque :

$$a(x' + ym') \equiv ax' + aym' \equiv b + a'ym \equiv b \bmod m .$$

5. Si $g \mid b$ et $(ax \equiv b) \bmod m$ alors $(x \equiv x') \bmod m'$.
Alors $(a'x \equiv b') \bmod m'$ et on sait que la solution est unique modulo m' . □

Exemple 80 *La congruence :*

$$(15x \equiv 3) \bmod 10$$

n'a pas de solution car $\text{pgcd}(15, 10) = 5 \nmid 3$. D'autre part, la congruence :

$$(15x \equiv 5) \bmod 10 ,$$

a plusieurs solutions dont une est celle de la congruence :

$$(3x \equiv 1) \bmod 2 ,$$

à savoir $x = 1$; et les autres ont la forme $(1 + k2)$ pour $k \in \mathbf{Z}$, donc : $x \in \{1, 3, 5, 7, 9\}$.

L'algorithme d'Euclide permet de calculer le pgcd de façon efficace.

Proposition 81 *Soient $a, b \in \mathbf{Z}$. Alors :*

$$\text{pgcd}(a, b) = \begin{cases} |a| & \text{si } b = 0 \\ \text{pgcd}(|b|, a \bmod |b|) & \text{autrement.} \end{cases} \quad (\text{A.2})$$

PREUVE. Si $b = 0$ alors on procède par analyse de cas sur a en se rappelant que $\text{pgcd}(0, 0) = 0$. Si $b \neq 0$ on sait (proposition 75) que :

$$a = q|b| + (a \bmod |b|) . \quad (\text{A.3})$$

On montre que d divise a et b ssi il divise $|b|$ et $a \bmod |b|$. En effet, si d divise b alors il divise $|b|$. L'équation (A.3) se réécrit comme :

$$a + (-q)|b| = a \bmod |b| .$$

Et donc d divise $a \bmod |b|$. D'autre part, si d divise $|b|$ alors il divise b . Et par l'équation (A.3) il divise a aussi. On a donc montré que si $b \neq 0$ alors les couples (a, b) et $(|b|, a \bmod |b|)$ ont les mêmes diviseurs. □

Exemple 82 *L'équation A.2 suggère un algorithme (efficace) pour calculer le pgcd. Par exemple,*

$$\text{pgcd}(-100, -35) = \text{pgcd}(35, 30) = \text{pgcd}(30, 5) = \text{pgcd}(5, 0) = 5 .$$

| Pas 0 | Pas 1 | Pas $k \geq 1$ for $r_k \neq 0$ |
|-------------|---------------------------------|---|
| $r_0 = a $ | $r_1 = b $ | $r_{k+1} = r_{k-1} \bmod r_k$ |
| | $q_1 = \lfloor r_0/r_1 \rfloor$ | $q_{k+1} = \lfloor r_k/r_{k+1} \rfloor$ |
| $x_0 = 1$ | $x_1 = 0$ | $x_{k+1} = q_k x_k + x_{k-1}$ |
| $y_0 = 0$ | $y_1 = 1$ | $y_{k+1} = q_k y_k + y_{k-1}$ |

Quand $r_k = 0$ on retourne : $\text{pgcd}(a, b) = r_{k-1}$, $x = (-1)^{k-1} x_{k-1}$ et $y = (-1)^k y_{k-1}$.

TABLE A.1 – Algorithme d'Euclide modifié

| k | 0 | 1 | | | |
|-------|-----|----|----|---|---|
| r_k | 100 | 35 | | | |
| q_k | - | 2 | | | |
| x_k | 1 | 0 | | | |
| y_k | 0 | 1 | | | |
| k | 0 | 1 | 2 | | |
| r_k | 100 | 35 | 30 | | |
| q_k | - | 2 | 1 | | |
| x_k | 1 | 0 | 1 | | |
| y_k | 0 | 1 | 2 | | |
| k | 0 | 1 | 2 | 3 | |
| r_k | 100 | 35 | 30 | 5 | |
| q_k | - | 2 | 1 | 6 | |
| x_k | 1 | 0 | 1 | 1 | |
| y_k | 0 | 1 | 2 | 3 | |
| k | 0 | 1 | 2 | 3 | 4 |
| r_k | 100 | 35 | 30 | 5 | 0 |
| q_k | - | 2 | 1 | 6 | |
| x_k | 1 | 0 | 1 | 1 | |
| y_k | 0 | 1 | 2 | 3 | |

TABLE A.2 – Application Euclide modifié avec $a = 100$ et $b = 35$ en 4 étapes

Par Bezout (proposition 77), on sait que pour tout $a, b \in \mathbf{Z}$ on peut trouver $x, y \in \mathbf{Z}$ tels que $\text{pgcd}(a, b) = ax + by$. Il est possible de généraliser l'algorithme d'Euclide pour qu'il calcule les coefficients x, y . La généralisation est décrite à la table A.1. Pour montrer sa correction on peut vérifier la récurrence (détails omis) :

$$r_k = (-1)^k x_k a + (-1)^{k+1} y_k b .$$

L'algorithme d'Euclide est efficace et on peut montrer qu'il termine en $O(\min(n, m))$, où \min est le minimum et n et m sont le nombre de bits nécessaires à représenter a et b , respectivement.

Un exemple d'application de l'algorithme avec $a = 100$ et $b = 35$ est décrite dans la table A.2.

Exercice 83 Appliquez l'algorithme d'Euclide modifié pour déterminer x, y tels que $\text{pgcd}(91, 143) = 91x + 143y$.

A.3 Nombres premiers

Définition 84 (nombre premier) Un nombre naturel $p > 1$ est premier s'il admet comme seuls diviseurs positifs 1 et p .

Chaque nombre naturel $a > 1$ admet une décomposition unique comme produit de nombres premiers.

Proposition 85 1. Chaque nombre $a > 1$ a un diviseur premier.

2. Si un premier divise un produit de nombres alors il divise au moins un des facteurs.
3. Si un nombre premier p divise un produit de nombres premiers $q_1 \cdots q_k$ alors il est égal au moins à un des facteurs.
4. Chaque entier $a > 1$ peut être écrit comme un produit de nombres premiers. Cette représentation est unique à permutation de facteurs près.

PREUVE.

1. Immédiat si a est premier. Sinon soit $p > 1$ le plus petit entier qui divise a ; p doit être premier, car si $b \mid p$ et $p > b > 1$ alors $b \mid a$.
2. On suppose $p \mid ab$ et $p \nmid a$. Comme p est premier, $\text{pgcd}(p, a) = 1$. Donc, $\exists x, y$ ($1 = ax + py$). Ainsi $b = abx + pby$. Mais $p \mid abx$ et $p \mid pby$ implique $p \mid b$.
3. Si $k = 1$ alors $p = q_1$. Si $k > 1$ alors par (2) ou bien $p \mid q_1$ ou bien $p \mid q_2 \cdots q_k$ et l'hypothèse de récurrence s'applique.
4. Pour l'existence, on procède par récurrence sur a . Immédiat pour $a = 2$ et pour $a > 2$ premier. Si $a > 2$ n'est pas premier alors il existe p premier tel que $1 < p < a$ et $p \mid a$. On applique l'hypothèse de récurrence à a/p .
Pour l'unicité, on procède aussi par récurrence sur a . Immédiat pour $a = 2$. Si $a = p_1 \cdots p_k = q_1 \cdots q_l$ alors p_1 doit être égal à un des q_i et l'hypothèse de récurrence s'applique à q/p_1 . \square

Il est facile de voir qu'on dispose d'une infinité de nombres premiers.

Proposition 86 L'ensemble des nombres premiers est dénombrable.

PREUVE. Par contradiction. Si p_1, \dots, p_n sont tous les nombres premiers alors posons :

$$p = p_1 \cdots p_n + 1 .$$

p ne peut pas être premier car $p > p_i$ pour $i = 1, \dots, n$. D'autre part, si p n'est pas premier il doit être divisible par un nombre premier. Mais ceci est aussi impossible car $p \bmod p_i = 1$. \square

Il est plus difficile de montrer que les nombres premiers sont abondants. Ce fait dont la preuve est omise, est connu comme le *théorème des nombres premiers* dont voici un énoncé assez précis.

Fait 87 Soient $n \geq 2$ et $\pi(n) = \#\{p \mid p \text{ premier et } p \leq n\}$. Alors pour $n \geq 17$ on a :

$$n/\log(n) \leq \pi(n) \leq (1,3)(n/\log(n)) .$$

A.4 Congruences et structure des entiers modulo

Définition 88 (congruence) Soit $a, b, m \in \mathbf{Z}$ avec $m \geq 1$. On dit que a est congru à b modulo m et on écrit $(a \equiv b) \bmod m$ si $m \mid (a - b)$ (ou de façon équivalente $m \mid (b - a)$).

Proposition 89 1. La relation $(- \equiv -) \bmod m$ est une relation d'équivalence.

2. Montrez que les conditions suivantes sont équivalentes :

(i) $(a \equiv b) \bmod m$, (ii) $a = b + km$, et (iii) $a \bmod m = b \bmod m$.

PREUVE.

1. Réflexivité : $m \mid 0$. Symétrie : $m \mid (a - b)$ implique $m \mid (b - a)$. Transitivité : si $m \mid (a - b)$ et $m \mid (b - c)$ alors $m \mid (a - c)$.
2. (i) implique (ii). $\exists k \quad km = (a - b)$ implique $a = b + km$. (ii) implique (iii). $a \bmod m = b + km \bmod m = b \bmod m$. (iii) implique (i). On suppose $r = a \bmod m = b \bmod m$, $a = q_1m + r$, $b = q_2m + r$. Alors $(a - b) = (q_1 - q_2)m$. \square

Notation Dans certains textes, on manipule les classes d'équivalence d'entiers modulo m . Dans ces notes de cours, on préfère prendre un représentant par classe d'équivalence, à savoir les éléments de l'ensemble $\mathbf{Z}_m = \{0, \dots, m - 1\}$.

L'ensemble \mathbf{Z}_m équipé avec les opérations d'addition et de multiplication (modulo m) est un anneau.

Proposition 90 Pour tout m entier, $m \geq 1$ la structure $(\mathbf{Z}_m, +, \cdot)$ est un anneau.

PREUVE. On commence par remarquer que si $(a \equiv b) \bmod m$ et $(c \equiv d) \bmod m$ alors : (i) $(-a \equiv -b) \bmod m$, (ii) $(a + c \equiv b + d) \bmod m$, et (iii) $(ac \equiv bd) \bmod m$. L'entier 0 est l'identité de l'addition modulaire. Par ailleurs, l'addition modulaire est associative, commutative et chaque élément a un élément inverse par rapport à l'addition. A savoir, l'inverse de a modulo m est $m - a$. La multiplication modulaire est aussi associative et commutative avec 1 comme élément neutre. Enfin, on peut vérifier la propriété de distributivité de l'addition par rapport à la multiplication. A savoir, dans \mathbf{Z}_m : $a(b + c) = (ab) + (ac)$. \square

On vient de voir que tout élément de \mathbf{Z}_m a une inverse additive. La proposition suivante caractérise les éléments de \mathbf{Z}_m qui ont aussi une inverse multiplicative.

Proposition 91 La congruence $(ax \equiv 1) \bmod m$ a une solution ssi $\text{pgcd}(a, m) = 1$, c'est-à-dire, a et m sont premiers entre eux. En plus, la solution, si elle existe, est unique.

PREUVE. On montre d'abord que si la congruence a une solution alors a et m sont premiers entre eux. Soit $(ax \equiv 1) \bmod m$, c'est-à-dire $(ax - 1) = km$. Soit $g = \text{pgcd}(a, m)$. Comme $1 = km - ax$, $g \mid m$ et $g \mid a$, on dérive $g \mid 1$, c'est-à-dire $g = 1$. On montre maintenant que si a et m sont premiers entre eux alors la congruence a une solution. Si $g = 1$ alors $\exists x, y \quad 1 = ax + my$. Donc x est une solution de la congruence. Il reste à montrer l'unicité de la solution. Soit $(av \equiv 1) \bmod m$ une autre solution. Alors $(ax \equiv av) \bmod m$. C'est-à-dire $m \mid (a(x - v))$. Comme $m \nmid a$, il suit que $m \mid (x - v)$. Donc $(x \equiv v) \bmod m$. \square

Remarque 92 L'algorithme d'Euclide modifié peut être utilisé pour calculer une inverse multiplicative. En effet si $\text{pgcd}(a, m) = 1$ alors on trouve x, y tels que $1 = ax + my$. Donc $m \mid (ax - 1)$.

On rappelle qu'un corps est un anneau dans lequel tous les éléments non-nuls ont une inverse multiplicative.

Corollaire 93 L'anneau \mathbf{Z}_m est un corps ssi m est premier.

PREUVE. Si m est premier alors chaque $a \in \mathbf{Z}_m \setminus \{0\}$ a la propriété que $\text{pgcd}(a, m) = 1$. D'autre part, si pour tout $a \in \mathbf{Z}_m \setminus \{0\}$ $\text{pgcd}(a, m) = 1$ alors m est premier. \square

A.5 Exponentiation rapide

Soit $(G, \cdot, 1_G)$ un *monoïde* avec une opération binaire \cdot qui est associative et commutative et un élément neutre 1_G . Si $g \in G$ et e un *entier positif*, on définit l'exposant g^e par :

$$g^0 = 1_G \quad g^{e+1} = g^e \cdot g .$$

La définition suggère une méthode de calcul de l'exposant qui demande un nombre d'opérations exponentiel dans le nombre de bits nécessaires à représenter e . Il se trouve qu'on peut faire beaucoup mieux !

Proposition 94 (carré itéré) *Soit $k + 1$ le nombre de bits dans la représentation en base 2 de e . Alors on peut calculer g^e avec au plus $2k$ opérations.*

PREUVE. Soit

$$e = \sum_{i=0, \dots, k} e_i 2^i ,$$

l'expansion binaire de e . Les coefficients e_i appartiennent à l'ensemble $\{0, 1\}$ avec $e_k = 1$ si $e > 0$. On observe :

$$\begin{aligned} g^e &= g^{\sum_{i=0, \dots, k} e_i 2^i} \\ &= \prod_{i=0, \dots, k} (g^{2^i})^{e_i} \\ &= \prod_{0 \leq i \leq k, e_i = 1} (g^{2^i}) . \end{aligned}$$

Pour le calcul de g^e on suit la procédure suivante. D'abord, on calcule g^{2^i} pour $0 \leq i \leq k$. En remarquant que :

$$g^{2^{i+1}} = (g^{2^i})^2 .$$

Pour ce faire, k multiplications (calculs du carré) sont nécessaires. Ensuite, on obtient g^e comme le produit des g^{2^i} tels que $e_i = 1$. Dans cette phase, au plus k multiplications sont nécessaires. On peut donc calculer g^e avec au plus $2k$ multiplications. \square

Si le monoïde est l'ensemble des nombres naturels avec la multiplication le calcul de l'exponentiel reste exponentiel dans la taille de g et e . La raison est qu'en général le nombre de bits nécessaires pour représenter le résultat d'une multiplication est environ deux fois le nombre de bits nécessaires à représenter les facteurs. Cette situation peut vraiment se présenter car si $g \approx 2^n$ et $e \approx 2^n$ alors $g^e \approx 2^{n^{2^n}}$. Donc le nombre de bits nécessaires à représenter le résultat de l'exponentiation est exponentiel dans le nombre de bits utilisés pour représenter g et e . Ainsi, tant qu'on utilise une représentation standard des nombres, tout algorithme pour calculer la fonction exponentielle prendra un temps exponentiel juste pour écrire le résultat du calcul.

La situation change radicalement si l'on travaille dans \mathbf{Z}_m . Dans ce cas, on peut toujours représenter le résultat d'une multiplication avec autant de bits qu'il en faut pour représenter m . On obtient donc un algorithme pour calculer $(g^e) \bmod m$ qui est polynomial en la taille de g , e et m . Par exemple, si la taille de g , e et m est n et si on utilise l'algorithme standard pour multiplier deux nombres, on obtient un algorithme pour le calcul de l'exposant modulaire qui est $O(n^3)$. Un tel algorithme peut traiter les grands nombres (avec un millier de chiffres) qu'on trouve dans les applications cryptographiques.

Exercice 95 Calculez $6^{73} \bmod 100$.

A.6 Restes chinois

Considérez le système de congruences :

$$(x \equiv 5) \bmod 7 \quad (x \equiv 3) \bmod 11 \quad (x \equiv 10) \bmod 13 .$$

Le théorème des restes chinois affirme que ce système a une solution unique modulo $7 \cdot 11 \cdot 13$ (laquelle ?) et nous donne une méthode efficace pour la trouver qui est résumée dans la table A.3.

Entrée : $(x \equiv a_i) \bmod m_i, i = 1, \dots, n, m_i$ premiers entre eux.
 Calcul : $M = m_1 \cdots m_n, \quad M_i = M/m_i, \quad y_i = (M_i^{-1}) \bmod m_i, \quad i = 1, \dots, n.$
 Sortie : $x = (\sum_{i=1, \dots, n} a_i y_i M_i) \bmod M.$

TABLE A.3 – Solution d'un système de congruences

Proposition 96 (restes chinois) Soient m_1, \dots, m_n des nombres entiers positifs premiers entre eux et soient a_1, \dots, a_n des nombres entiers. Alors le système de congruences :

$$(x \equiv a_i) \bmod m_i, i \in \{1, \dots, n\}$$

a une solution x qui est unique modulo $M = m_1 \cdots m_n$.

PREUVE. On considère d'abord l'unicité. Si x et x' sont deux solutions du système alors $(x \equiv x') \bmod m_i$ pour $i = 1, \dots, n$, par transitivité de la congruence. Comme les m_i sont premiers entre eux, on dérive que $(x \equiv x') \bmod M$. En effet, si $(x - x') = km_1 = k'm_2$ alors les facteurs dans m_1 doivent être dans k' .

Pour l'existence, on pose :

$$M_i = M/m_i, \quad i = 1, \dots, n.$$

On remarque que m_i et M_i sont premiers entre eux. Par l'algorithme d'Euclide, on détermine y_i tels que :

$$(M_i y_i \equiv 1) \bmod m_i, \quad i = 1, \dots, n.$$

Ensuite, on pose :

$$x = (\sum_{i=1, \dots, n} a_i y_i M_i) \bmod M.$$

On dérive :

$$(a_i y_i M_i \equiv a_i) \bmod m_i.$$

Comme m_i divise M_j pour $i \neq j$ on a $(a_j y_j M_j \equiv 0) \bmod m_i$. Ainsi :

$$(x \equiv a_i y_i M_i + \sum_{j \neq i, j=1, \dots, n} a_j y_j M_j \equiv a_i) \bmod m_i, \quad i = 1, \dots, n.$$

□

Exercice 97 Déterminez x tel que :

$$(x \equiv 5) \bmod 7, \quad (x \equiv 3) \bmod 11 \quad \text{et} \quad (x \equiv 10) \bmod 13.$$

Exercice 98 Montrez que sans l'hypothèse que les modules sont premiers entre eux on n'a pas l'unicité de la solution modulo M .

Un corollaire du théorème des restes chinois est que l'anneau produit $\mathbf{Z}_m \times \mathbf{Z}_n$ est isomorphe à l'anneau $\mathbf{Z}_{m \cdot n}$ si $\text{pgcd}(m, n) = 1$. Plus en général, soient m_1, \dots, m_n des nombres positifs. Alors on peut construire l'ensemble produit :

$$\mathbf{Z}_{m_1} \times \cdots \times \mathbf{Z}_{m_n} \tag{A.4}$$

et on peut équiper cet ensemble avec des opérations de somme et de produit qui sont l'extension par composante des opérations somme et produit disponibles pour $\mathbf{Z}_{m_i}, i = 1, \dots, n$. Le lecteur peut vérifier que la structure obtenue est un anneau qu'on appelle *anneau produit*.

Proposition 99 Si les m_1, \dots, m_n sont premiers entre eux alors l'anneau produit (A.4) est isomorphe à l'anneau \mathbf{Z}_M où $M = m_1 \cdots m_n$.

PREUVE. On vérifie que la fonction :

$$i : \mathbf{Z}_M \rightarrow \mathbf{Z}_{m_1} \times \cdots \times \mathbf{Z}_{m_n}, \quad i(x) = (x \bmod m_1, \dots, x \bmod m_n),$$

est un isomorphisme d'anneaux. La surjectivité correspond à l'existence d'une solution d'un système de congruences et l'injectivité à l'unicité (modulo M). \square

Remarque 100 *Le théorème des restes chinois permet aussi de paralléliser les opérations arithmétiques sur \mathbf{Z}_M . En effet, si $M = m_1 \cdots m_n$ on peut calculer avec un degré de parallélisme n en $\mathbf{Z}_{m_1}, \dots, \mathbf{Z}_{m_n}$ et à la fin du calcul on peut retrouver le résultat dans \mathbf{Z}_M .*

A.7 Groupe multiplicatif

On collecte un certain nombre de résultats sur les groupes finis et notamment sur le groupe multiplicatif des entiers modulo m qu'on définit dans la suite.

Définition 101 (groupe multiplicatif) *Soit $m \geq 1$ entier. On dénote par \mathbf{Z}_m^* l'ensemble des entiers $n \in \mathbf{Z}_m$ qui ont une inverse multiplicative. En d'autres termes :*

$$\mathbf{Z}_m^* = \{a \in \mathbf{Z}_m \mid \text{pgcd}(a, m) = 1\}.$$

Exercice 102 *Vérifier que \mathbf{Z}_m^* avec l'opération de multiplication modulo m forme un groupe commutatif.*

Définition 103 (fonction d'Euler) *La fonction d'Euler ϕ est définie par :*

$$\phi(m) = \#\mathbf{Z}_m^* \quad m \in \mathbf{Z}, m \geq 1.$$

Exercice 104 1. Montrez que \mathbf{Z}_m^* est un groupe abélien par rapport à la multiplication dans \mathbf{Z}_m .

2. Calculez \mathbf{Z}_{12}^* .

3. Calculez $\phi(m)$ pour $m = 1, \dots, 15$.

4. Quelle est la valeur de $\phi(p)$ pour p premier ?

Le théorème des restes chinois permet de dériver la proposition suivante.

Proposition 105 *Si m_1, \dots, m_n sont des nombres entiers positifs et premiers entre eux alors :*

$$\phi(m_1 \cdots m_n) = \phi(m_1) \cdots \phi(m_n).$$

PREUVE. On réutilise la preuve de la proposition 99 pour montrer que $\mathbf{Z}_{m_1}^* \times \cdots \times \mathbf{Z}_{m_n}^*$ est isomorphe à \mathbf{Z}_M^* où $M = m_1 \cdots m_n$. \square

Soit G un groupe abélien fini avec unité 1. Si $g \in G$ alors g^{-1} dénote son inverse. Pour tout $a \in \mathbf{Z}$ on peut aussi définir l'exposant g^a par :

$$g^0 = 1 \quad g^a = gg^{(a-1)} \quad g^{-a} = (g^{-1})^a \text{ où } a > 0.$$

Définition 106 *L'ordre de $g \in G$, écrit $\text{ord}(g)$, est le plus petit nombre positif e tel que $g^e = 1$.*

Exercice 107 *Expliquez pourquoi l'ordre est bien défini (un e positif existe toujours) et déterminez l'ordre de 2 dans \mathbf{Z}_{13}^* .*

On peut observer que $g^e = 1$ si et seulement si e est un multiple de $\text{ord}(g)$.

Proposition 108 Soit $g \in G$ et $e \in \mathbf{Z}$. Alors $g^e = 1$ ssi $\text{ord}(g) \mid e$.

PREUVE. (\Leftarrow) On suppose $n = \text{ord}(g)$ et $e = kn$. Alors :

$$g^e = g^{kn} = (g^n)^k = 1 .$$

(\Rightarrow) On suppose $g^e = 1$ et $e = qn + r$ avec $0 \leq r < n$ et $n = \text{ord}(g)$. Alors :

$$g^r = g^{e-qn} = g^e (g^n)^{-q} = 1 .$$

Mais comme n est le plus petit entier positif tel que $g^n = 1$, il faut que $r = 0$. □

Exercice 109 Soit $g \in G$ et $k, l \in \mathbf{Z}$. Alors $g^l = g^k$ ssi $(l \equiv k) \bmod \text{ord}(g)$.

La définition de l'exposant suggère que si l'on considère toutes les puissances d'un élément $g \in G$ on obtient un sous-groupe de G . On parle du sous-groupe généré par g et l'ensemble des puissances est aussi appelé *orbite*.

Exercice 110 Soit $g \in G$. Montrez que : $\langle g \rangle = \{g^a \mid a \in \mathbf{Z}\}$ est un sous-groupe de G et que $\text{ord}(g) = \# \langle g \rangle$.

Le théorème de Lagrange en théorie des groupes assure que la cardinalité d'un sous-groupe d'un groupe fini divise la cardinalité du groupe.

Proposition 111 (Lagrange) Soit H un sous-groupe d'un groupe fini G . Alors $\#H \mid \#G$.

PREUVE. Soit H un sous-groupe. On écrit $a \sim b$ si $ab^{-1} \in H$. On vérifie que \sim est une relation d'équivalence et que si $c \in H$ alors $ca \sim a$.

Si $a \in G$ alors $[a] = \{c \mid ca^{-1} \in H\}$ dénote sa classe d'équivalence. On remarque que $[1] = H$. Toutes les classes d'équivalence ont la même cardinalité car la fonction $f : [1] \rightarrow [a]$ définie par $f(c) = ca$ est bijective. Par exemple, pour la surjectivité :

$$\begin{aligned} b \in [a] &\Rightarrow a \sim b \\ &\Rightarrow ab^{-1} \in H \\ &\Rightarrow ba^{-1} \in H \\ &\Rightarrow f(ba^{-1}) = b \end{aligned}$$

Ainsi la cardinalité de H divise la cardinalité de G . □

L'application du théorème de Lagrange au groupe multiplicatif \mathbf{Z}_m^* permet de dériver un résultat connu comme *petit théorème de Fermat* qui est utilisé dans la définition du système à clef publique RSA (chapitre 8).

Corollaire 112 (Fermat) Soit m entier positif et $a \in \mathbf{Z}_m^*$. Alors :

$$(a^{\phi(m)} \equiv 1) \bmod m .$$

PREUVE. Par la proposition 111 (Lagrange), on sait que $\phi(m)$, la cardinalité du groupe multiplicatif, est un multiple de la cardinalité du groupe généré par a . Donc il y a un entier k tel que :

$$\# \mathbf{Z}_m^* = \phi(m) = k \cdot \text{ord}(a) = k \cdot \# \langle a \rangle .$$

Il suit que : $(a^{\phi(m)} \equiv (a^{\text{ord}(a)})^k \equiv 1^k \equiv 1) \bmod m$. □

A.8 Test de primalité

Le corollaire 112 suggère une méthode efficace pour *tester* la primalité d'un nombre $n \geq 2$. Choisir $a \in \{2, \dots, n-1\}$ et vérifier :

$$(a^{n-1} \equiv 1) \pmod{n} \quad (\text{A.5})$$

par *exponentiation rapide*. Notez que le test dépend du choix de a . Si le nombre n est premier, il passe le test pour tout choix de a . D'autre part, si pour un choix de a le nombre n ne passe pas le test alors on est sûr qu'il n'est pas premier. Il reste à estimer la probabilité qu'un nombre qui n'est pas premier passe le test. Si cette probabilité est au plus $\epsilon < 1$ alors en répétant le test k fois (avec des choix indépendants de a) on aura que la probabilité de passer les k tests est au plus ϵ^k qui tend vers 0.

La proposition suivante va dans la bonne direction car elle montre que le test échoue chaque fois qu'on tombe sur un nombre qui n'est pas premier avec n .

Proposition 113 *Si a n'est pas premier avec n alors $(a^{n-1} \not\equiv 1) \pmod{n}$.*

PREUVE. Si $\text{pgcd}(a, n) \neq 1$ alors $\exists k \geq 2 \ a = kx, n = ky$. Comme $n-1 \geq 1$ on a aussi $k \mid a^{n-1}$. Si $(a^{n-1} \equiv 1) \pmod{n}$ on pourrait dériver $k(x+y) = 1$ avec $k \geq 2$, ce qui est impossible. \square

Définition 114 (nombre de Carmichael) *Un nombre composé $n \geq 2$ est un nombre de Carmichael si pour tout $a \in \mathbb{Z}_n$ premier avec n on a :*

$$(a^{n-1} \equiv 1) \pmod{n}.$$

On sait qu'il y a une infinité de nombres de Carmichael et qu'ils sont rares. Le plus petit nombre de Carmichael est :

$$3 \cdot 11 \cdot 17 = 561.$$

Exercice 115 *Estimez la probabilité de choisir un élément $a \in \{2, \dots, 560\}$ qui ne passe pas le test de Fermat, à savoir $(a^{560} \not\equiv 1) \pmod{561}$.*

On peut montrer que si n n'est pas premier et n'est pas un nombre de Carmichael alors avec probabilité au moins $1/2$ on tombe sur un a qui ne passe pas le test de Fermat. Dans ce cas, si l'on répète le test 100 fois la probabilité de déclarer un nombre composé premier est au plus 2^{-100} . Il est possible de concevoir des tests de primalité encore plus efficaces et qui n'ont pas de difficultés avec les nombres de Carmichael. La table A.4 présente un test dû à Miller et Rabin [Mil76, Rab80].

Exemple 116 *On prend le plus petit nombre de Carmichael : $n = 561 = 3 \cdot 11 \cdot 17$. Alors $n-1 = 560 = 2^4 \cdot 35$. Donc $k = 4$ et $m = 35$. On choisit $a = 2 \in \{1, \dots, 560\}$ et on calcule :*

$$\begin{aligned} (2^{35} \equiv 263) \pmod{561}, & \quad (2^{2 \cdot 35} \equiv 166) \pmod{561}, \\ (2^{4 \cdot 35} \equiv 67) \pmod{561}, & \quad (2^{8 \cdot 35} \equiv 1) \pmod{561}. \end{aligned}$$

Ceci est assez pour conclure que 561 est composé.

On va montrer que tout nombre premier passe le test de Miller-Rabin. On a d'abord besoin d'étudier les solutions d'une congruence quadratique.

Proposition 117 *Soit $p \geq 3$ premier. La congruence*

$$(x^2 \equiv 1) \pmod{p}$$

a exactement deux solutions modulo p . A savoir, 1 et -1 .

Entrée : $n \geq 3$, impair et
 $k \geq 1$, m impair et $(n-1) = 2^k m$.

Test : choisir $a \in \{2, \dots, n-1\}$ et vérifier :
 $(a^m \equiv 1) \pmod n$ ou $(a^m \equiv -1) \pmod n$ ou
 $(a^{2^m} \equiv -1) \pmod n$ ou \dots ou $(a^{2^{(k-1)}m} \equiv -1) \pmod n$.

TABLE A.4 – Test de Miller-Rabin

PREUVE. Clairement 1 et -1 sont deux solutions différentes car $p \geq 3$. Soit maintenant x une solution. Alors :

$$p \mid (x-1) \text{ ou } p \mid (x+1),$$

et donc $(x \equiv 1) \pmod p$ ou $(x \equiv -1) \pmod p$. \square

Proposition 118 *Tout nombre premier $n \geq 3$ passe le test de Miller-Rabin.*

PREUVE. On procède *par contradiction*. On suppose que $n \geq 3$ premier *échoue le test*. Ceci veut dire que :

$$(a^m \not\equiv 1) \pmod n \text{ et } (a^m \not\equiv -1) \pmod n \text{ et } \dots \text{ et } (a^{2^{(k-1)}m} \not\equiv -1) \pmod n.$$

Comme n est *premier*, on sait que $\phi(n) = n-1$. En sachant $(n-1) = 2^k m$, par le petit théorème de Fermat :

$$(a^{(n-1)} \equiv a^{2^k m} \equiv 1) \pmod n.$$

Il suit que $a^{2^{k-1}m}$ est une *racine carrée* de 1 modulo n . Par la proposition 117, il y a *deux solutions possibles* ($+1$ ou -1) et comme le *test échoue*, on doit avoir :

$$(a^{2^{k-1}m} \equiv 1) \pmod n.$$

Maintenant, $a^{2^{k-2}m}$ est une *racine* de 1 modulo n . Par un raisonnement similaire on peut conclure que :

$$(a^{2^{k-2}m} \equiv 1) \pmod n.$$

On itère cet argument jusqu'à $k=1$, pour obtenir :

$$(a^m \equiv 1) \pmod n,$$

ce qui *contredit* l'hypothèse que n échoue le test. \square

Fait 119 *Si un nombre n est impair et composé alors l'ensemble $\{1, \dots, n-1\}$ contient au plus $(n-1)/4$ nombres qui sont premiers avec n et qui ne témoignent pas du fait que n est composé.*

En d'autres termes, si on répète le test de Miller-Rabin k fois (de façon indépendante) la probabilité de donner une réponse erronée est au plus $1/4^k$. Ce qui veut dire que pour avoir une probabilité d'erreur comparable il faut faire deux tests de Fermat pour chaque test de Miller-Rabin.

A.9 Groupes cycliques et générateurs

Définition 120 *On dit qu'un groupe est cyclique s'il y a un élément g qui génère le groupe, c'est-à-dire tel que $G = \langle g \rangle$.*

Il y a une relation surprenante entre la *fonction d'Euler* et le *nombre de générateurs*.

Proposition 121 Soit G un groupe fini cyclique. Alors G a exactement $\phi(\#G)$ générateurs.

PREUVE. Si $\text{pgcd}(k, n) = 1$ alors g^k est un générateur. En effet, par Bezout $\exists x, y \quad kx + ny = 1$. Tout élément de G a la forme g^m pour $0 \leq m < n$ et on remarque $g^m = g^{(kx+ny)m} = g^{kxm} g^{nym} = (g^k)^{xm}$. On note aussi que si g^k est un générateur alors $\text{pgcd}(k, n) = 1$. Car on doit avoir $g^{kx} = g^1$. Donc $kx \equiv 1 \pmod n$. Donc $\text{pgcd}(k, n) = 1$. On peut donc conclure que G a exactement $\phi(\#G)$ générateurs à savoir : $\{g^k \mid 1 \leq k < n, \text{pgcd}(k, n) = 1\}$. \square

De plus, *cyclicité* et *primalité* sont connectées. Pour illustrer cette connexion on utilise le fait suivant.

Fait 122 Soit p premier et soit $P(x)$ un polynôme non-nul dans une variable x de degré d . Alors :

$$\#\{y \in \mathbf{Z}_p \mid (P(y) \equiv 0) \pmod p\} \leq d .$$

On dit qu'un polynôme non-nul de degré d a au plus d racines distinctes.¹ En particulier, en prenant $p(x) = x^d - 1$ on déduit que :

$$\#\{y \in \mathbf{Z}_p \mid (y^d \equiv 1) \pmod p\} \leq d . \quad (\text{A.6})$$

Proposition 123 Si p est premier alors le groupe \mathbf{Z}_p^* est cyclique. Et par la proposition 121 il admet exactement $\phi(p-1)$ générateurs.

PREUVE. Le groupe multiplicatif \mathbf{Z}_p^* hérite de \mathbf{Z}_p la propriété (A.6). On va montrer que tout sous-groupe G de \mathbf{Z}_p^* (donc en particulier \mathbf{Z}_p^*) a la propriété suivante :

$$\#\{y \in G \mid (y^d \equiv 1) \pmod p\} \leq d . \quad (\text{A.7})$$

On procède par récurrence sur $n = \#G$.

- $n = 1$. 1 est un générateur.
- n est premier. On sait que pour tout $x \in G$, $\text{ord}(x)$ doit diviser n et comme n est premier on a que $\text{ord}(x) = n$ pour tout $x \neq 1$.
- n est de la forme q^c pour q premier et $c \geq 2$. Par contradiction. S'il n'y a pas de générateur dans G alors pour tout $x \in G$, $\text{ord}(x)$ divise q^{c-1} . Mais on sait que le polynôme $P(y) = y^{q^{c-1}}$ a au plus q^{c-1} racines dans G . Contradiction.
- $n = n_1 \cdot n_2$ avec $\text{pgcd}(n_1, n_2) = 1$. On définit :

$$H_i = \{x \in G \mid (x^{n_i} \equiv 1)\} \quad i = 1, 2 ,$$

et on vérifie que H_i est un sous-groupe de G qui par ailleurs hérite de G la propriété (A.7). Donc $\#H_i \leq n_i < n$ et l'hypothèse inductive s'applique pour conclure que H_i a un générateur g_i (pour $i = 1, 2$). On montre que $g_1 g_2$ est un générateur pour G . D'abord, on remarque que par la proposition 77 (Bezout) :

$$\exists x, y \quad n_1 x + n_2 y = 1 .$$

et donc pour tout $g \in G$ on a :

$$(g \equiv g^{n_1 x + n_2 y}) \pmod p .$$

Par ailleurs, comme $\#G = n_1 n_2$ on a :

$$((g^{x n_1})^{n_2} \equiv (g^{y n_2})^{n_1} \equiv 1) \pmod p ,$$

1. Ceci est vrai pour tout corps, donc en particulier pour un polynôme sur \mathbf{Z}_p .

ce qui implique que :

$$(g^{x_{n_1}}) \in H_2 \quad (g^{y_{n_2}}) \in H_1 .$$

Ainsi, comme les g_i sont des générateurs pour H_i , on doit avoir i_1, i_2 tels que :

$$(g \equiv g^{n_1 x + n_2 y} \equiv (g^{x_{n_1}})(g^{y_{n_2}}) \equiv g_1^{i_1} g_2^{i_2}) \pmod{p} .$$

Il reste à trouver j tel que :

$$((g_1^{i_1} g_2^{i_2}) \equiv (g_1 g_2)^j) \pmod{p} .$$

Par la proposition 96 (restes chinois), on peut trouver j tel que :

$$(j \equiv i_1) \pmod{n_1} \quad (j \equiv i_2) \pmod{n_2} .$$

En d'autres termes, il y a des entiers k_1, k_2 tels que :

$$i_1 = j + k_1 n_1 = i_2 = j + k_2 n_2 .$$

On a donc :

$$\begin{aligned} g_1^{i_1} g_2^{i_2} &\equiv (g_1^{j+k_1 n_1} g_2^{j+k_2 n_2}) \\ &\equiv (g_1^j) (((g_1)^{n_1})^{k_1}) (g_2^j) (((g_2)^{n_2})^{k_2}) \\ &\equiv (g_1^j) (g_2^j) \\ &\equiv (g_1 g_2)^j \pmod{p} . \end{aligned}$$

□

Exercice 124 Trouvez tous les générateurs de \mathbf{Z}_{13}^* .

Dans certaines applications cryptographiques, on a besoin de déterminer le générateur d'un groupe Z_p^* pour p premier. On peut suivre une approche similaire à celle utilisée pour trouver un grand nombre premier. D'abord on s'assure que les générateurs sont abondants dans Z_p^* et ensuite on conçoit un test efficace pour savoir si un élément de \mathbf{Z}_p^* est un générateur.

Proposition 125 (recherche générateur) Si $p > 2$ est premier et $a \in \mathbf{Z}_p^*$ alors a est un générateur du groupe multiplicatif si et seulement si $(a^{(p-1)/r} \not\equiv 1) \pmod{p}$ pour tout r premier tel que $r \mid (p-1)$. En particulier, si l'on sait que $(p-1)/2$ est premier alors on peut tester de façon efficace si a est un générateur.

PREUVE. Par Lagrange (proposition 111), on sait que $\text{ord}(a) \mid (p-1)$. Donc si $\text{ord}(a) < (p-1)$ alors la décomposition en facteurs premiers de $\text{ord}(a)$ doit contenir au moins un facteur premier en moins que $(p-1)$. Si en plus $q = (p-1)/2$ est premier alors $\phi(p-1)$ se décompose en $2 \cdot q$. Donc a est un générateur ssi $(a^2 \not\equiv 1) \pmod{p}$ et $(a^q \not\equiv 1) \pmod{p}$. □

Remarque 126 On sait que les générateurs sont assez abondants dans \mathbf{Z}_p^* car on peut montrer que pour tout p premier il y a au moins de l'ordre de \sqrt{p} générateurs. Mais cela ne suffit pas ! Il faut aussi savoir tester de façon efficace si un élément $a \in \mathbf{Z}_p^*$ est un générateur ; d'où l'intérêt de prendre p tel que $(p-1)/2$ est premier. On note aussi que l'abondance de nombres premiers permet de trouver de façon assez rapide un tel p .

Définition 127 (logarithme discret) Soit p premier, g générateur pour \mathbf{Z}_p^* . La fonction logarithme discret en base g est une fonction sur $\text{dlog}_g : \mathbf{Z}_p^* \rightarrow \mathbf{Z}_p^*$ définie par :

$$\text{dlog}_g(A) = a \quad \text{si } a \in \{1, \dots, p-1\} \text{ et } (g^a \equiv A) \pmod{p} .$$

De façon équivalente, le logarithme discret en base g est la fonction inverse de la fonction exponentielle en base g :

$$\exp_g : \mathbf{Z}_p^* \rightarrow \mathbf{Z}_p^* \quad \exp_g(a) = (g^a) \pmod{p} .$$

En effet pour $A, a \in \mathbf{Z}_p^*$ on a que $\text{dlog}_g(A) = a$ ssi $(g^a \equiv A) \pmod{p}$. L'algorithme du carré itéré (proposition 94) nous permet de calculer la fonction exponentielle de façon efficace. Par contre, on ne connaît pas une méthode efficace pour calculer le logarithme discret dans \mathbf{Z}_p^* . La fonction exponentielle modulo p est donc candidate à être une fonction à sens unique (définition 38).

Annexe B

Algèbre linéaire

Dans ce chapitre, on rappelle la définition de déterminant et de matrice inverse. Soit R un anneau commutatif avec un élément 0 comme unité de l'addition et un élément 1 comme unité de la multiplication. Pour fixer les idées, on peut prendre comme anneau \mathbf{Z}_m , les entiers modulo m . Soit $R[m, n]$ la collection des matrices sur R avec m lignes et n colonnes, avec éléments génériques A, B, \dots . Si A est une matrice alors on dénote avec $A[i, j]$ l'élément avec coordonnées (i, j) . L'addition et la multiplication de matrices est définie de façon standard. L'ensemble $R[n, n]$ des matrices carrées de dimension n possède une structure d'anneau.

Définition 128 (déterminant) *Le déterminant d'une matrice $A \in R[n, n]$ est défini par récurrence :*

$$\det(A) = \begin{cases} A[1, 1] & \text{si } n = 1 \\ \sum_{j=1, \dots, n} (-1)^{i+j} A[i, j] \cdot \det(A_{i,j}) & \text{si } n > 1, \end{cases}$$

où $A_{i,j} \in R[n-1, n-1]$ est la matrice qui résulte de A en effaçant la ligne i et la colonne j .

On sait que le déterminant ne dépend pas du choix de la ligne (ou colonne) $i \in \{1, \dots, n\}$.

Exemple 129 *On calcule le déterminant d'une matrice 2×2 . On suppose*

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

Alors en calculant par rapport à la première ligne :

$$A_{1,1} = [a_{2,2}] \quad A_{1,2} = [a_{2,1}] .$$

Donc :

$$\det(A) = (-1)^{1+1} \cdot a_{1,1} \cdot A_{1,1} + (-1)^{1+2} \cdot a_{1,2} \cdot A_{1,2} = a_{1,1} \cdot a_{2,2} - a_{1,2} \cdot a_{2,1} .$$

Exercice 130 *Donnez une formule pour calculer le déterminant d'une matrice 3×3 .*

Définition 131 (matrice adjointe) *Soit $A \in R[n, n]$. Alors la matrice adjointe $\text{adj}(A) \in R[n, n]$ est définie par :*

$$\text{adj}(A)[i, j] = (-1)^{i+j} \det(A_{j,i}) .$$

Fait 132 *Soit $A \in R[n, n]$. L'inverse multiplicative de A existe si et seulement si $\det(A)$ a une inverse (multiplicative) dans R et dans ce cas :*

$$A^{-1} = (\det(A))^{-1} \cdot \text{adj}(A) .$$

Exercice 133 On prend $R = \mathbf{Z}_{11}$ et

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Calculez l'inverse de A .

Corollaire 134 Soient $A \in R[n, n]$, $b \in R[n, 1]$ et $F(x) = A \cdot x + b$ une fonction affine sur R^n . La fonction F a une inverse si et seulement si $\det(A)$ a une inverse en R . En particulier, si $R = \mathbf{Z}_m$ alors F est inversible si et seulement si $\text{pgcd}(\det(A), m) = 1$.

Remarque 135 Pour calculer le déterminant et l'inverse multiplicative de façon efficace on peut utiliser des méthodes qui raffinent l'élimination de Gauss (solution de systèmes d'équations linéaires).

Annexe C

Travaux pratiques et dirigés

C.1 Utilisation de Gnu Privacy Guard

GnuPG est un logiciel *libre* qui met en oeuvre différents aspects de la cryptographie symétrique et asymétrique. L'objectif de ce travail pratique est d'apprendre à utiliser le logiciel GnuPG. Le logiciel est disponible à l'adresse <http://www.gnupg.org/> et il est décrit dans le manuel GnuPG disponible à la page <http://www.gnupg.org/documentation/>.

Questions Voici certains points qui vous devriez aborder.

1. Étude des fonctionnalités de base du logiciel (par exemple, voir premier chapitre du manuel). Vous devriez être capable de chiffrer un message et/ou de le signer.
2. GNUPG offre de nombreuses possibilités d'interface avec d'autres logiciels (par exemples avec des programmes pour la gestion du courrier électronique). Voir http://www.gnupg.org/related_software/. Vous êtes invités à explorer certaines de ces possibilités.
3. GNUPG adopte un modèle pour la gestion des clefs qui est basé sur une *toile de confiance* (voir manuel). Vous devez être capable d'expliquer ce mécanisme.

C.2 Cryptoanalyse de chiffrements affines

Le but de ce TP est de se familiariser avec les programmes disponibles sur le site <http://www.apprendre-en-ligne.net/crypto/menu/index.html> afin d'analyser des textes chiffrés avec des systèmes affines.

Vigenère Considérez le texte chiffré suivant dont on sait que le texte source est en anglais et que la méthode de chiffrement utilisée est celle de Vigenère.

```
ADSBG VLDQZ CXTZB BJAEL TMVIY XAQWU PLTHB KDTSG SVNPK KCWSN OHFBC
JZNRB ECIFM XAZAX HLEJT UGCRS BBKES UUFLB SPJNW PDQOE ABMPK TIWET
WNKTZ YOTBH SVIAF GQZZF MEGPA AVAET TEOQA YDVFI YJKTG IIOIQ FBXTV
WXXMF CULWE PTEMS UEHRL ZFEPU SGAFS RJCYY QGULA OAYOI JDJXS RMLDR
ECNSW YWROF RSSUT WGXTG ENIND TJCCG BMNRA JSGQC WFHLE AOYAK PTDXO
PWLYF TMPYI ZCSDC UAFMC BCBQO AYDJA BSLJB UWTCT QWIDH MEZXH POADG
ECWNP PEMEV FOESI MBSML TGWZS ZEYOW TTRHF IYOQW WHAGE EQNLT ZEWGK
PVRFB SZCZV MDZSN QAZXG LSPFT TQALY EAMCT MPGR L BTYOV OYGST OQBSP
KHDWA QTRJB SPJTY IRJJB TVZTA XBS
```

Questions

1. Les sites proposent différentes méthodes pour retrouver le texte clair et la clef. Par exemple, la méthode de Babbage/Kasiski et celle basée sur l'indice de coïncidence. Pouvez-vous appliquer ces méthodes au texte en question ?

Produit de Permutation et Vigenère Le texte ci-dessous a été obtenu par un système cryptographique qui est un produit d'un système par permutation et d'un système de Vigenère (la notion de système produit est décrite dans l'exercice D.2).

Questions (suite)

2. Considérez un chiffrement par permutation (on dit aussi transposition) d'un texte français. Y-a-t-il une relation entre l'indice de coïncidence du texte clair et du texte chiffré ?
3. Pouvez-vous retrouver le texte d'origine dont on sait qu'il est en français ?

```
RGCRC WOEPD FIQXS OPPDQ VREIV LRRZG OIRAC IHWQR DMOPG OWYLW
EREPD PGRGI XGPZ TUWGS TELVU EWTUM KQFIG WGUWI OERPP TOIGT
FSXVN ETQHQ QIXTO RTEWE EKGGL TUMGL YVHRC OEWRP PLTJD YFAYW
LRGML YQIVD LVDRF EWRRR GPPIG HRLFW RYPEA HUXGE WEUFC LPXHM
PSLZD VEEGI SWNED EDKPE XYVSO NPXGY TTPVW WGSZY EHGUI EHEWX
DVEVG QTPBE GULXU WDLZG UHGML YEVGL EMVEP LNIVX NABYL IVOXF
HENAC WQZGR WIVYT ESIUF GVPVW WKLDI QXGVL ERYKL PQRVV ODXSV
QCSIU JNAYH HVCRA RVYTU SIUSE EPFLW CIWIQ PDAET HYUQF WMIPE
TEXVC EDHLV GLPTH GIRTR VIOAT FQIUE PVHGJ BYMVI TRTWH HGBZR
QMUSL EWIPL LMUWU CYHOI XOXXH YQIEU OMNNP WDFTI LIUER ADMOP
PPWEL RVBTI UXGNO XHQGN AIQWG NLVOE KTLPX EOEXI DLFUC GQHCL
ESPQG OFIVF VNYIH XUATR VIFAY SWSPP FQPIC OOSUW PNEHH IQMMV
UYUED YHPKQ PIVYP DOIQX PEDEL VVPTI VVGDF IDRID EWDMP LMEVM
EEOIV YJEGI RBFEX KQWGI DRHYT DLMQX FEYMV IWVPX GQGNE RHWC
CXHQC NTILM CEDXS XUJFW GISUP MHWRA PIQWV EASVW EDPRW IGSEH
WGJRP ULIPS OYHXW TZMVI PRPLR UQRPI YYGDP WDWVE PXHVT CDEYI
CTZMM WWRML XGQNB ULWGS PYHXX AYXFL SRWIV EWILP HFCRM VIPGU
XMHPG UAIUI RRPWW EWIDW URVEE RLGJE EIVSK IUEPE GSWIS GGDFR
GSWAC MTYGD AIXRG TPVPI UTPTD VCLLJ WEPCL LHPNE CSQXG DZPDR
SSPRW QWEWE HSTTW TQXTE PVHRF SEHDR VLLXH IGEEP UGQEF IOYKD
OWFIP OOIVW RUDYQ ZKNTU UEEOF XDRVE SWXVN VPVEI UECXH RCLWS
DKGPW WFEPT FSXWN ETISI KEESO XHAYI UVGGL ZGEPT LIUWN DRVDR
CEPWS MINPE DRUIQ TLXKL BEUGG EFMOZ GUEUX PEHLV QIOAR SHIVT
DYVWG ADSOH UTDHH QQNLV VIGDT YHRVQ ZIOIP OMPHG UMEIH XVMZV
QIPCZ ETYGR MRWMN CLXVE GOFPS YFEES SWUED WHGJE FIWTQ RCPHY
URPQL SUIZR DJHRP WGMGU YSQKC XEBAB
```

C.3 Vernam faible

On utilise des clés de 8 octets qu'on décompose en 4 groupes de 2 octets $k_0k_1k_2k_3$. Le chiffrement d'une séquence de $n+1$ `char` Java x_0, \dots, x_n (un `char` Java est sur 2 octets) est défini par la séquence y_0, \dots, y_n où : $y_i = x_i \oplus k_{(i \bmod 4)}$ (il y a un opérateur en Java pour faire un xor bit à bit).

Questions

1. Programmez en Java les fonctions : G pour générer une clef avec une distribution uniforme, E pour chiffrer et D pour déchiffrer. Il doit être possible de visualiser les textes clairs et chiffrés et de tester rapidement ces fonctions.
2. Proposez une méthode de cryptanalyse qui étant donné un texte chiffré qui correspond à un texte clair en langue naturelle (anglais ou français par exemple), calcule, avec une probabilité de succès significative, la clef et le texte clair.

3. Mettez en oeuvre votre méthode de cryptanalyse et déterminez de façon expérimentale la longueur minimale d'un texte qui permet à votre programme de mener à bien l'analyse avec une probabilité au moins 1/2.

C.4 Mise en gage d'un bit

Un protocole de mise en gage d'un bit (*bit commitment*) réalise l'idée suivante :

- Alice écrit une valeur $b \in \{0, 1\}$ sur un papier et met le papier dans une boîte dont elle est la seule à avoir la clef.
- La phase de mise en gage consiste à donner la boîte à Bob.
- La phase de révélation consiste à donner la clef à Bob.

En suivant [Nao91], on peut utiliser un générateur pseudo-aléatoire pour réaliser cette idée. Alice et Bob fixent un générateur pseudo-aléatoire

$$\mathcal{G} : 2^n \rightarrow 2^{3n}$$

qui transforme n bits en $3n$ bits. Si Alice et Bob ne cherchent pas à tricher le protocole se passe de la façon suivante :

- Bob tire $R \in 2^{3n}$ avec probabilité uniforme et l'envoie à Alice.
- Alice tire $b \in 2$ et $Y \in 2^n$ avec probabilité uniforme, elle calcule :

$$R' = (\text{if } b \text{ then } \mathcal{G}(Y) \text{ else } \mathcal{G}(Y) \oplus R) \in 2^{3n}$$

et elle envoie R' à Bob. C'est la phase de mise en gage.

- Plus tard, Alice envoie Y à Bob. C'est la phase de révélation.

Questions

1. Pour tricher Bob peut choisir un R à sa guise et ensuite essayer de déterminer la valeur de b à partir de R' . Bob a-t-il une chance significative de réussir ?
2. Pour tricher Alice peut envoyer un R' tel que

$$(\exists Y \ R' = \mathcal{G}(Y)) \vee (\exists Y \ R' = \mathcal{G}(Y) \oplus R)$$

dans la phase de mise en gage et ensuite produire un Y approprié dans la phase de révélation. Alice a-t-elle des chances significatives de réussir ?

3. Proposez un protocole pour jouer à pile ou face au téléphone (section 1.6) qui utilise un protocole de mise en gage.
4. Soit $\mathcal{G}^i(Y)$ le i -ème bit de la séquence $\mathcal{G}(Y)$ et $\mathcal{G}^{\leq i}$ les premiers i bits de la séquence $\mathcal{G}(Y)$, où $1 \leq i \leq 3n$. On considère la variante suivante du protocole de mise en gage :
 - Mise en gage : Alice tire $b \in 2$ et $Y \in 2^n$ et transmet à Bob $\mathcal{G}^{\leq 3n-1}(Y)$ et $\mathcal{G}^{3n}(Y) \oplus b$.
 - Révélation : Alice transmet à Bob Y qui peut donc calculer $\mathcal{G}(Y)$ et vérifier les valeurs transmises dans la phase précédente.

Supposons que Alice trouve $Y_1, Y_2 \in 2^n$ tels que $\mathcal{G}(Y_1) \neq \mathcal{G}(Y_2)$ et $\mathcal{G}^{\leq 3n-1}(Y_1) = \mathcal{G}^{\leq 3n-1}(Y_2)$ (ce qui est compatible avec la définition de PRG). Montrez que dans ce cas Alice peut tricher.

C.5 Réseaux de substitution permutation

On considère un chiffrement à bloc :

$$\mathcal{F}_k : 2^{16} \rightarrow 2^{16}$$

basé sur un réseau de substitutions et de permutations qu'on abrège en RSP.

Substitution On utilise la première ligne de la première boîte de DES qui est une fonction $S : 2^4 \rightarrow 2^4$ spécifiée par le tableau suivant :

| | | | | | | | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Entrée : | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| Sortie : | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |

Permutation Comme permutation on utilise une fonction $P : \{1, \dots, 16\} \rightarrow \{1, \dots, 16\}$ qui est spécifiée par le tableau suivant :

| | | | | | | | | | | | | | | | | |
|----------|---|---|---|----|---|---|----|----|---|----|----|----|----|----|----|----|
| Entrée : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Sortie : | 1 | 5 | 9 | 13 | 2 | 6 | 10 | 14 | 3 | 7 | 11 | 15 | 4 | 8 | 12 | 16 |

Clef Une clef K est aussi sur 16 bits. La clef K_i du tour i pour $i = 1, 2, 3, 4$ est obtenue en effectuant $4 * (i - 1)$ décalages circulaires vers la droite de la clef K . Ainsi si $K = x_1 \cdot x_2 \cdot x_3 \cdot x_4$ et $x_i \in 2^4$ pour $i = 1, 2, 3, 4$, on aura :

$$\begin{aligned} K_1 &= x_1 \cdot x_2 \cdot x_3 \cdot x_4 \\ K_2 &= x_4 \cdot x_1 \cdot x_2 \cdot x_3 \\ K_3 &= x_3 \cdot x_4 \cdot x_1 \cdot x_2 \\ K_4 &= x_2 \cdot x_3 \cdot x_4 \cdot x_1 \end{aligned}$$

Tour Le chiffrement se décompose en 4 tours. Le tour i , pour $i = 1, 2, 3, 4$ consiste à transformer un bloc $x = (x_1 \cdot x_2 \cdot x_3 \cdot x_4)$, où $x_i \in 2^4$ comme suit :

$$\begin{aligned} (y_1 \cdot y_2 \cdot y_3 \cdot y_4) &= (x \oplus K_i) && \text{(XOR avec la clef)} \\ (w_1 \cdot w_2 \cdot w_3 \cdot w_4) &= (S(y_1) \cdot S(y_2) \cdot S(y_3) \cdot S(y_4)) && \text{(Substitution)} \\ z &= P(w_1 \cdot w_2 \cdot w_3 \cdot w_4) && \text{(Permutation)} \end{aligned}$$

Questions

1. Programmez les fonctions de chiffrement et de déchiffrement à bloc : \mathcal{F}_k et \mathcal{F}_k^{-1} .
2. Considérez \mathcal{F} comme un PRF (*pseudo-random function generator*) pour des fonctions de type $2^{16} \rightarrow 2^{16}$. Pour traiter des textes de longueur arbitraire, programmez les fonctions de chiffrement et déchiffrement en mode compteur (*counter mode*).
3. Supposons qu'on chiffre les messages m_i , $i = 1, \dots, \ell$ où chaque message est composé de 100 caractères. On aura donc pour $m_i = m_{i,1} \dots m_{i,100}$:

$$E_k(m_i) = [r \leftarrow 2^{16} : (r, m_{i,1} \oplus \mathcal{F}_k(r+1), \dots, m_{i,100} \oplus \mathcal{F}_k(r+100))]$$

où l'addition est effectuée modulo 2^{16} . Déterminez de façon expérimentale une valeur de ℓ qui, étant donné le chiffrement d'un texte de 100 caractères, permet de calculer (au moins) un caractère du texte clair (presque toujours).

C.6 Programmation de RSA en Java

L'objectif est de se familiariser avec la programmation en Java d'un certain nombre de fonctions cryptographiques. Les API Java contiennent un certain nombre de classes qui permettent de représenter aisément des concepts cryptographiques. Voir, par exemple, la classe `BigInteger`

<http://docs.oracle.com/javase/7/docs/api/java/math/BigInteger.html>

et plus en général l'extension Java `cryptography extension`

<http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>

Questions

1. Programmez :
 - Un générateur de nombres pseudo-aléatoires.
 - Un testeur de primalité type Miller-Rabin.
 - Une méthode de factorisation en nombre premiers.
 - Les fonctions de chiffrement et de déchiffrement RSA.
2. Déterminez le plus petit nombre premier sur 512 bits.
3. Factorisez : 831802500 et 18533588383.
4. Chiffrez et déchiffrez avec RSA.
5. Programmez une méthode de chiffrement/déchiffrement asymétrique hybride qui combine la méthode RSA avec le PRF \mathcal{F} programmé dans le cadre des réseaux de substitution-permutation. Notez qu'on peut utiliser le PRF comme un PRG et définir :

$$E_{pk}^{hyb}(m_1 \cdots m_\ell) = [k \leftarrow 2^{16} : (E_{pk}^{RSA}(k), \mathcal{F}_k(1) \oplus m_1, \dots, \mathcal{F}_k(\ell) \oplus m_\ell)] .$$

Il est conseillé de lire l'exercice D.11. Préparez une démonstration à la machine.

C.7 Routage anonyme

Dites moi avec qui vous parlez et je vous dirai qui vous êtes. En plus de protéger la confidentialité et l'intégrité des messages il faut aussi rendre anonyme la structure des communications. On imagine l'expérience suivante.

- On a n participants p_1, \dots, p_n avec clefs publiques authentifiées k_1, \dots, k_n et n sites s_1, \dots, s_n . L'état initial du système est décrit par une permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ qui affecte chaque participant à un site et une fonction $f : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ qui précise l'interlocuteur de chaque participant. Ainsi $\pi(i) = j$ si et seulement si le participant p_i se trouve au site j et $f(i) = j$ si et seulement si le participant p_i souhaite envoyer 1 message au participant j .
- Un attaquant observe tous les messages échangés entre les sites mais il ne peut pas déchiffrer les messages chiffrés avec les clefs publiques des participants. Par exemple, supposons qu'on a 3 participants p_1, p_2, p_3 qui se trouvent aux sites s_1, s_2, s_3 et qui souhaitent communiquer les messages m_1, m_2, m_3 d'après la fonction :

$$f(1) = 2, f(2) = 1, f(3) = 1 .$$

L'attaquant pourra observer une permutation arbitraire des 3 communications :

$$(s_1, m_1, s_2), (s_2, m_2, s_1), (s_3, m_3, s_2)$$

où l'on dénote par le triplet (s, m, s') l'origine du message (s), le message (m) et la destination du message (s').

- On dit qu'un protocole garantit l'*anonymat* si les traces observables par l'attaquant sont invariantes par permutation des participants dans les sites. Considérons l'exemple ci-dessus. Même si les messages m_1, m_2, m_3 sont vides, l'anonymat n'est pas garanti car les traces observables changent si l'on permute, par exemple, le site de p_2 avec le site de p_3 .
- Pour essayer de confondre l'attaquant, on introduit un serveur S avec clef publique k_S et on envisage de faire passer tous les messages par S . De plus on envisage d'utiliser de la cryptographie à clef publique.

Questions

1. Proposez un protocole qui va utiliser le serveur S pour essayer d'obtenir une communication anonyme. Supposons que i veut envoyer un message m à j . Vous devez préciser la forme des messages que i envoie à S et que S envoie à j . Ce faisant, vous devez aborder les questions suivantes :
 - Quelles sont vos hypothèses sur la taille des messages ?
 - Quelles sont vos hypothèses sur l'ordre temporel de traitement des messages par le serveur ?
 - Combien de messages faut-il échanger pour compléter l'envoi des n messages originaux ?
 - Que se passe-t-il si deux messages envoyés sont identiques ?
2. En principe un participant j peut recevoir un message anonyme m d'un autre participant i . Modifiez votre protocole de façon à que chaque participant ait la possibilité de répondre à un message anonyme. On fera l'hypothèse qu'un participant i à l'origine d'un message peut insérer dans m de l'information (par exemple une clef de session) qui permettra à j de répondre sans pour autant dévoiler l'identité de i .

Remarque 136 *Un problème avec l'approche décrite est que maintenant le serveur centralise toutes les conversations et sait exactement qui parle avec qui. Un pas vers la distribution consiste à introduire 2 serveurs S_1 et S_2 . Maintenant un message passe de i à S_1 , ensuite de S_1 à S_2 et enfin de S_2 à sa destination finale. Ceci peut être généralisé à k serveurs et ces serveurs peuvent être choisis de façon aléatoire (voir par exemple [Cha81]). On parle de routage à oignon (onion routing) car le message est encapsulé dans une suite de chiffrements qui sont éliminés au fur et mesure que le message progresse vers sa destination finale.*

C.8 Pseudo-primalité

On sait que si un nombre n est *premier* alors pour tout $a \in \{2, \dots, n-1\}$ on a $(a^{n-1} \equiv 1) \bmod n$ (test de Fermat). On dit qu'un nombre $n \geq 2$ est un *pseudo-premier* si :

$$\forall a \in (Z_n)^* \quad (a^{n-1} \equiv 1) \bmod n$$

L'ensemble des *témoins* pour n est défini par :

$$T_n = \{a \in (Z_n)^* \mid (a^{n-1} \not\equiv 1) \bmod n\}$$

On définit aussi $T_n^c = (Z_n)^* \setminus T_n$. Notez que n est pseudo-premier si et seulement si il n'a pas de témoin ($T_n = \emptyset$).

Questions Le but est de montrer qu'on peut construire un programme probabiliste efficace pour décider si un nombre est pseudo-premier. Soit $n \geq 2$. Montrez que :

1. Si $a \in \mathbf{Z}_n \setminus \mathbf{Z}_n^*$ alors $(a^{n-1} \not\equiv 1) \bmod n$.
2. Si $a \in T_n$ et $d \in T_n^c$ alors $((ad) \bmod n) \in T_n$.
3. Si $a \in T_n$, $d, d' \in T_n^c$ et $d \neq d'$ alors $(ad \not\equiv ad') \bmod n$.

4. Conclure que si n n'est pas pseudo-premier alors au moins la moitié des éléments de $(\mathbb{Z}_n)^*$ sont des témoins pour n .
5. Dérivez un algorithme probabiliste efficace qui décide si un nombre $n \geq 2$ est pseudo-premier. Vous pouvez supposer que $\phi(n) \geq \sqrt{n}$ et votre algorithme peut donner une réponse erronée avec une probabilité au plus 2^{-100} .
6. Programmez en Java l'algorithme en utilisant la classe `BigInteger`.

C.9 Calcul du logarithme discret

Soit p premier et g un générateur pour le groupe multiplicatif $(\mathbb{Z}_p)^*$ dont l'ordre est $p-1$. Soit $t = \lfloor \sqrt{p} \rfloor$. On cherche à concevoir un programme pour calculer le logarithme discret en base g dont le temps de calcul est $O(t \log t)$ en supposant que le calcul d'une multiplication modulo p prend $O(1)$. Le programme prend en entrée $y \in (\mathbb{Z}_p)^*$ et calcule l'unique $x \in \mathbb{Z}_{p-1}$ tel que $(g^x \equiv y) \bmod p$.

Questions Dans la suite les calculs se font modulo p .

1. Donnez la complexité asymptotique en fonction de t d'un programme qui calcule et trie la séquence :

$$g^0, g^t, g^{2 \cdot t}, \dots, g^{t \cdot (t+1)}$$

2. Montrez qu'en $O(t \log t)$ on peut calculer la séquence :

$$y \cdot g^0, y \cdot g, \dots, y \cdot g^{(t-1)}$$

et ensuite déterminer i et k tels que :

$$(y \cdot g^i \equiv g^{k \cdot t}) \tag{C.1}$$

3. Expliquez comment utiliser l'équation (C.1) pour calculer x .
4. Programmez l'algorithme et comparez sa performance en pratique avec le programme naïf qui calcule la séquence :

$$g^0, g^1, \dots, g^{p-2}$$

jusqu'à trouver x tel que $(g^x \equiv y) \bmod p$. Préparez une démonstration à la machine.

C.10 Signature aveugle

On analyse un protocole de *signature aveugle*. On prend p, q nombres premiers, un module $n = p \cdot q$, un exposant de chiffrement e , et un exposant de déchiffrement d avec les hypothèses standards sur RSA ; \mathbb{Z}_n^* dénote le groupe multiplicatif.

Le participant A utilise ce système RSA pour signer des messages.¹ Le participant B souhaite obtenir la signature d'un document $m \in \mathbb{Z}_n^*$ sans pour autant dévoiler la valeur de m . Pour ce faire, B choisit avec une probabilité uniforme une valeur $r \in \mathbb{Z}_n^*$ et calcule : $m' = (m \cdot r^e) \bmod n$. Ensuite B envoie à A la valeur m' , et reçoit en réponse $(m'^d) \bmod n$.

Questions

1. Montrez que la fonction $r \mapsto (r^e) \bmod n$ est une permutation sur \mathbb{Z}_n^* .
2. Montrez que si $m \in \mathbb{Z}_n^*$ alors $m' \in \mathbb{Z}_n^*$.
3. Montrez que pour chaque choix de $m, m' \in \mathbb{Z}_n^*$ il existe unique $r' \in \mathbb{Z}_n^*$ tel que $(m \equiv r' m') \bmod n$.

1. On utilise la version naïve de la signature RSA sans fonction d'hachage.

4. Soit $P(m)$ la probabilité que le message clair soit m (on suppose $P(m) > 0$ pour tout m). Soit $P(m \mid m')$ la probabilité conditionnelle que le message clair soit m en sachant que le message à signer de façon aveugle est m' . Peut-on déduire que $P(m \mid m') = P(m)$?
5. La réponse à la question précédente, change-t-elle si le message m n'est pas premier avec n ?
6. On suppose maintenant que A utilise le même système RSA pour signer de façon aveugle et pour recevoir des messages. On suppose aussi que B intercepte un message $(m^e) \bmod n$ destiné à A . Comment B peut-il obtenir de A le déchiffrement du message ?

Annexe D

Exercices

D.1 Générateur affine

Le texte chiffré $c = 111\ 101\ 100\ 101\ 101\ 010\ 111$ a été obtenu en effectuant un xor du texte clair p avec une suite z de bits générés en utilisant un générateur basé sur la congruence linéaire :

$$(z_{i+1} \equiv az_i + b) \bmod 8.$$

Les entiers z_i ainsi générés sont interprétés comme des nombres à trois bits $b_{i2}b_{i1}b_{i0}$ de façon telle que $z_i = b_{i2} \cdot 2^2 + b_{i1} \cdot 2 + b_{i0}$. Ainsi on aura :

$$\begin{aligned} z &= b_{02}b_{01}b_{00} \ b_{12}b_{11}b_{10} \ \cdots \ b_{62}b_{61}b_{60} \\ c &= p \oplus z \end{aligned}$$

On connaît les premiers 9 bits du texte clair p :

$$p = 100\ 111\ 001 \ \cdots$$

Déterminez le reste du texte clair.

D.2 Combinaison de systèmes cryptographiques

Pour améliorer la sécurité, une idée naturelle est de *combinaison* les systèmes cryptographiques. Comment les combiner et la combinaison est-elle utile ? Une construction standard consiste à prendre le *produit* de deux systèmes cryptographiques. Soient $\mathcal{P} = \mathcal{C}$ et

$$S_i = (\mathcal{P}, \mathcal{P}, \mathcal{K}_i, G_i, E_i, D_i) \quad i = 1, 2,$$

deux systèmes cryptographiques. On définit leur produit par :

$$S_1 \times S_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, (G_1, G_2), E, D),$$

avec :

$$E_{(k_1, k_2)}(p) = E_{k_2}(E_{k_1}(p)), \quad D_{(k_1, k_2)}(c) = D_{k_1}(D_{k_2}(c)).$$

Par exemple, soit S un système par décalage (César) et L un système linéaire défini comme un système affine avec constante additive $b = 0$. Alors le système affine peut être vu comme $L \times S$:

$$E_{(a,b)}(x) = ((ax) \bmod m + b) \bmod m = (ax + b) \bmod m$$

où $\text{pgcd}(a, m) = 1$.

Questions

1. On considère le système produit $S \times L$. Le système résultant est-il équivalent à un système connu ?
2. Montrez que le produit de systèmes cryptographiques est associatif.

3. Montrez que le système par décalage et le système linéaire sont idempotents, *c'est-à-dire* $S \times S$ est équivalent à S et $L \times L$ à L .
4. Montrez que si deux systèmes sont idempotents et si leur produit commute alors leur produit est aussi idempotent.
5. Conclure qu'un système affine est idempotent.

D.3 Permutation et Vigenère jetable

1. Dans un système à permutation jetable, pour chiffrer un texte composé de n caractères on utilise une permutation sur $\{0, \dots, n-1\}$. Un tel système assure-t-il la confidentialité parfaite ?
2. Pour chiffrer un message composé de n caractères sur l'alphabet $\{A, B, \dots, Z\} = \mathbf{Z}_{26}$ on utilise un système de Vigenère sur \mathbf{Z}_{26} avec une clef aussi longue que le texte. Un tel système assure-t-il la confidentialité parfaite ?

D.4 Distance statistique et équivalence calculatoire

On dénote avec U_n une v.a.d. avec distribution uniforme sur n bits, à savoir si $w \in 2^n$ alors $P(U_n = w) = 1/2^n$. On dénote avec Y_n une v.a.d. calculée de la façon suivante :

$$Y_n = [x \leftarrow 2^n; y \leftarrow 2^n; \text{if } (x = y) \text{ then } 0^n \text{ else } x]$$

On tire n bits x , puis on tire n bits y , si x est identique à y on retourne 0^n (n fois 0), sinon on retourne x . On dénote avec $\{\mathcal{G}_n : 2^n \rightarrow 2^{2n}\}_{n \in \mathbf{N}}$ un PRG qui prend n bits et produit $2n$ bits.

1. Calculez $P(Y_n = w)$ pour $w \in 2^n$.
2. Calculez la distance statistique : $\Delta(U_n, Y_n)$.
3. Les familles $\mathbf{U} = \{U_n\}_{n \in \mathbf{N}}$ et $\mathbf{Y} = \{Y_n\}_{n \in \mathbf{N}}$ sont-elles équivalentes au sens calculatoire ?
4. Estimez la distance statistique : $\epsilon(n) = \Delta(\mathcal{G}_n \circ U_n, U_{2n})$.
5. La fonction $\epsilon(n)$ de la question 4, est-elle négligeable ?

D.5 PRG à sens unique

Soit $\{\mathcal{G}_n : 2^n \rightarrow 2^{2n}\}_{n \in \mathbf{N}}$ un PRG. Soit \mathcal{I} un algorithme PPT d'inversion qui associe n bits à $2n$ bits. Montrez que la fonction :

$$i(n) = P(k \leftarrow 2^n : \mathcal{G}_n(\mathcal{I}(\mathcal{G}_n(k))) = \mathcal{G}_n(k))$$

est négligeable. En d'autres termes, un PRG qui double la taille de son entrée est une fonction à sens unique.

D.6 Tolérance aux erreurs du mode CBC

On se place dans le cadre du chiffrement en mode CBC (section 5.5).

1. On a une *erreur de transmission* du premier bloc ce qui veut dire que l'émetteur calcule correctement le premier bloc c_1 mais le récepteur obtient un bloc $c'_1 \neq c_1$. Quels blocs seront déchiffrés correctement par le récepteur ?
2. On suppose maintenant que l'émetteur fait une *erreur de calcul* dans le deuxième bloc, ainsi il calcule un bloc c'_2 plutôt que c_2 et le transmet (correctement). Quels blocs chiffrés seront affectés par cette erreur ? Quels blocs de texte clair pourront être calculés par le récepteur, en supposant qu'il n'y ait pas d'erreur de transmission ?

D.7 Feistel faible

On utilise un schéma de Feistel où la clef de tour est toujours la même et la fonction de codage f_k est le xor avec k . On a donc :

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus (R_i \oplus k) \end{aligned}$$

Quelles sont les faiblesses de ce schéma en fonction du nombre de tours ?

D.8 Feistel et PRP

Si f_1, \dots, f_n sont des fonctions de type $2^n \rightarrow 2^n$ et $L_0, R_0 \in 2^n$ alors on définit $Feistel(f_1, \dots, f_n)(L_0 R_0)$, $n \geq 1$ comme la suite de $2n$ bits obtenus en appliquant la construction de Feistel à l'entrée $L_0 R_0$. En particulier on a :

$$\begin{aligned} Feistel(f_1)(L_0 R_0) &= R_0 \cdot (L_0 \oplus f_1(R_0)) \\ Feistel(f_1, f_2)(L_0 R_0) &= (L_0 \oplus f_1(R_0)) \cdot R_0 \oplus f_2(L_0 \oplus f_1(R_0)) \end{aligned}$$

1. Expliquez pourquoi la fonction $Feistel(f_1, \dots, f_n)$ est une permutation de type $2^{2n} \rightarrow 2^{2n}$.
2. Prenons le cas $n = 1$. Montrez que si l'on choisit f_1 avec probabilité uniforme il est possible de distinguer $Feistel(f_1)$ d'une permutation choisie avec probabilité uniforme.
3. Même question pour le cas $n = 2$.¹

D.9 Mac faibles

Soit $\mathcal{F} : 2^n \rightarrow [2^n \rightarrow 2^n]$ un PRF. Soit $m = m_1 \cdots m_\ell$ un message avec $m_i \in 2^n$ et ℓ polynomial en n . On considère 3 schémas pour construire un *Mac* :

1.

$$Mac_k^1(m) = \mathcal{F}_k(m_1) \oplus \cdots \oplus \mathcal{F}_k(m_\ell)$$

où \oplus est le ou-exclusif sur n bits.

2.

$$Mac_k^2(m) = [r \leftarrow 2^n : (r, \mathcal{F}_k(r) \oplus \mathcal{F}_k(m_1) \oplus \cdots \oplus \mathcal{F}_k(m_\ell))]$$

où la notation $r \leftarrow 2^n$ veut dire que r est un vecteur de n bits tiré avec probabilité uniforme.

3.

$$Mac_k^3(m) = [r \leftarrow 2^n : (r, \mathcal{F}_k(r) \oplus \mathcal{F}_k((1)_2 + m_1) \oplus \cdots \oplus \mathcal{F}_k((\ell)_n + m_\ell))]$$

où $(i)_2$ est la représentation en base 2 du nombre naturel i , $+$ est l'addition en base 2 et on suppose $\ell < 2^{n/2}$. Montrez qu'aucun de ces schémas est sûr dans le sens de la définition 52.

D.10 Substitution non-linéaire

La substitution S est une fonction qui transforme 4 bits en 4 bits dont on donne ci-dessous la définition (un chiffre hexadécimal = 4 bits) :

| | | | | | | | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Entrée : | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| Sortie : | E | 4 | D | 1 | 2 | F | B | 8 | 3 | A | 6 | C | 5 | 9 | 0 | 7 |

1. Montrez que cette fonction ne s'exprime pas comme une fonction affine sur $(\mathbf{Z}_2)^4$. Ceci veut dire qu'on ne peut pas trouver une matrice A , de dimension 4×4 , et un vecteur b , à coefficients dans \mathbf{Z}_2 , tels que :

$$(A(x_3, x_2, x_1, x_0) + b)_2 = S((x_3, x_2, x_1, x_0)_2) \quad \text{pour tout } (x_3, x_2, x_1, x_0) \in (\mathbf{Z}_2)^4.$$

Ici on dénote avec $(y_3, y_2, y_1, y_0)_2$ la valeur en base 2 de la suite de bits $y_3 y_2 y_1 y_0$, à savoir $\sum_{i=0, \dots, 3} y_i \cdot 2^i$.

2. Estimez le nombre de fonctions affines sur $(\mathbf{Z}_2)^4$ et comparez-le au nombre de permutations sur $(\mathbf{Z}_2)^4$. On rappelle que d'après la formule de Stirling :

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n.$$

1. Pour $n \geq 3$ on peut montrer que la construction donne effectivement une permutation pseudo-aléatoire.

D.11 RSA faible

1. Programmez une fonction Java d'en tête :

```
public static int racinecub(int x)
```

qui prend en entrée un entier positif x et qui retourne (s'il existe) un entier y tel que $y^3 = x$ et un entier -1 autrement. Il s'agit donc de calculer (si elle existe) la racine cubique dans \mathbf{Z} (et non pas dans \mathbf{Z}_n !). Votre programme devrait être linéaire dans le nombre de bits nécessaires à représenter x .

2. Rappelez les principes du chiffrement hybride à clef publique.

Supposons que pour implémenter le chiffrement hybride on dispose d'un système de chiffrement symétrique avec clef de 128 bits et d'un système RSA avec module $n = p \cdot q$ de 512 bits.

3. Sous quelles hypothèses peut-on choisir un exposant de chiffrement $e = 3$?
4. Supposons que pour le chiffrement RSA on choisit bien un exposant de chiffrement $e = 3$ et que pour le chiffrement hybride on regarde une clef symétrique k comme un entier dans l'intervalle $[0, 2^{128} - 1]$. Montrez que dans ce cas un attaquant a une méthode efficace pour déchiffrer les messages sans connaître la clef de déchiffrement.

D.12 Diffie-Hellman en groupe

Les participants sont identifiés par un élément de $\mathbf{Z}_n = \{0, \dots, n-1\}$. Les participants souhaitent établir une clef commune. Pour chaque couple de participants $(i, j) \in \mathbf{Z}_n^2$ on dispose d'un canal de communication authentifié mais qui n'assure pas la confidentialité des messages échangés. Les participants appliquent une généralisation du protocole de *Diffie-Hellman*.

- Le participant 0 communique à chaque participant un (grand) nombre premier p et un générateur g pour $(\mathbf{Z}_p)^*$.
- Chaque participant choisit de façon indépendante et avec probabilité uniforme un élément $a_i \in \{2, \dots, p-2\}$.
- A la fin du protocole, la clef commune est :

$$K = A_{\{0, \dots, n-1\}} = (g^{\prod_{i \in \{0, \dots, n-1\}} a_i}) \bmod p$$

1. Proposez un protocole d'échange de valeurs de la forme (I, A_I) :

$$(A_I \equiv g^{\prod_{i \in I} a_i}) \bmod p \quad I \subset \{0, \dots, n-1\}$$

avec les objectifs suivants :

- Permettre à chaque participant de calculer la clef K .
- Éviter que l'observation des valeurs échangées (I, A_I) permette à un attaquant de calculer la clef commune K .

Vous devez expliciter les échanges de valeurs pour le cas où $n = 2$ (trois participants).

D.13 Programmation fonction de hachage en Java

Si t_0, \dots, t_n est une suite de bits alors on dénote par $val(t_0, \dots, t_n)$ l'entier $\sum_{i=0, \dots, n} (t_i \cdot 2^i)$. Si m est un entier positif alors $\lfloor \log_2 m \rfloor$ dénote l'entier $\max\{k \mid 2^k \leq m\}$. Étant donné deux entiers positifs g et m , pour calculer le hachage d'une suite de bits t_0, \dots, t_n on calcule $h = (g^{val(t_0, \dots, t_n)}) \bmod m$ et on retourne les $\lfloor \log_2 m \rfloor$ bits les moins significatifs de la représentation binaire de h . Écrire une fonction Java d'en tête :

```
static boolean[] hash (boolean[] t, int g, int m)
```

qui implémente cette méthode. La méthode doit avoir une complexité proportionnelle à $(n+1)$ (la longueur de la suite de bits en entrée). Aucune fonction de bibliothèque est autorisée.

D.14 Signature RSA à anneau

On décrit une version simplifiée d'un système de signature qui permet à un ensemble d'utilisateurs $U = \{1, \dots, m\}$, $m \geq 2$ de signer un message de façon telle que :

- le récepteur du message peut vérifier que le message a bien été signé par un des utilisateurs sans savoir lequel et
- chaque utilisateur peut signer sans la coopération des autres utilisateurs.

Ce qui est demandé est simplement que chaque utilisateur ait une clef publique authentifiée.

Hypothèses

- Chaque utilisateur $i \in U$ a une clef publique RSA (e_i, n_i) et une clef privée RSA d_i où n_i est un modulo sur au plus s bits. On écrit $E_i(x)$ pour $(x^{e_i}) \bmod n_i$ et $D_i(y)$ pour $(y^{d_i}) \bmod n_i$.
- On fixe une fonction de hachage $h : 2^* \rightarrow 2^s$.
- Soit $t = 2^s$. On définit $E'_i : \mathbf{Z}_t \rightarrow \mathbf{Z}_t$ par

$$E'_i(x) = \begin{cases} n_i q + E_i(r) & \text{si } x = n_i q + r, 0 \leq r < n_i, n_i(q+1) \leq t \\ x & \text{autrement} \end{cases}$$

Signature pour $m = 2$ L'utilisateur $i = 1$ signe un message msg comme suit :

- Il choisit $x_2 \in \mathbf{Z}_t$.
- Il calcule $y_2 = E'_2(x_2)$.
- Il calcule $x_1 = D'_1(h(msg) \oplus y_2)$.
- La signature est $(\{1, 2\}, x_1, x_2)$.

Vérification pour $m = 2$ On vérifie que :

$$h(msg) = E'_1(x_1) \oplus E'_2(x_2)$$

Questions

1. Expliquez pourquoi $E'_i, i = 1, \dots, m$, est une permutation sur \mathbf{Z}_t et définissez la permutation inverse D'_i .
2. Expliquez pourquoi un message correctement signé passe la vérification.
3. Montrez que si $h(msg) = h(msg')$ alors toute signature pour msg est aussi une signature valide pour msg' .
4. Généralisez le schéma de signature et de vérification à $m > 2$ utilisateurs.

D.15 Test pour les résidus quadratiques

Soit $p > 2$ nombre premier. On dit que a dans le groupe multiplicatif $(\mathbf{Z}_p)^*$ est un résidu quadratique s'il existe $x \in (\mathbf{Z}_p)^*$ tel que $(x^2 \equiv a) \bmod p$. Rappel : $(\mathbf{Z}_p)^*$ est un groupe cyclique avec $\phi(p-1)$ générateurs.

1. Calculez les résidus quadratiques pour $p = 7$.
2. Montrez que si a est un résidu quadratique alors $(a^{(p-1)/2} \equiv 1) \bmod p$.
3. Montrez qu'un générateur de $(\mathbf{Z}_p)^*$ ne peut pas être un résidu quadratique et calculez les générateurs pour $p = 7$.
4. Montrez que si $a \in (\mathbf{Z}_p)^*$ et $(a^{(p-1)/2} \equiv 1) \bmod p$ alors a est un résidu quadratique.

D.16 Entiers de Blum

Un entier de Blum est un nombre entier n tel que $n = p \cdot q$, p et q sont deux nombres premiers différents et $(p \equiv q \equiv 3) \bmod 4$.

1. Trouvez le plus petit entier de Blum.
2. On sait qu'il y a un nombre infini de nombres premiers. Montrez qu'il y a un nombre infini de nombres premiers p tels que $(p \equiv 1) \bmod 4$ ou $(p \equiv 3) \bmod 4$.

On souhaite montrer qu'il y a une infinité de nombres premiers p tels que $(p \equiv 3) \bmod 4$. On *raisonne par l'absurde*, en supposant qu'il y a un nombre fini de nombres premiers $p_1 < \dots < p_n$ tels que $(p_i \equiv 3) \bmod 4$. On considère le nombre

$$m = (4p_1 \cdots p_n) - 1$$

3. Montrez que m est impair et que $(m \equiv 3) \bmod 4$.
4. Montrez qu'il doit y avoir un i tel que $1 \leq i \leq n$ et p_i divise m .
5. Dérivez une contradiction.
6. Conclure qu'il y a une infinité d'entiers de Blum.
7. Proposez un algorithme probabiliste qui prend en entrée un entier N et rend comme résultat (s'il termine) un couple p, q tel que $n = pq$, $n \geq N$ et n est un entier de Blum.

Bibliographie

- [And01] Ross Anderson. *Security engineering*. Wiley, 2001.
- [BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2) :364–383, 1986.
- [BS16] Dan Boneh and Victor Shoup. *A graduate course in applied cryptography*. Pre-print of a book in preparation, 2016.
- [Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2) :84–88, 1981.
- [CvHP91] David Chaum, Eugène van Heijst, and Birgit Pfizmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 470–484, 1991.
- [Dam89] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 416–427, 1989.
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6) :644–654, 1976.
- [DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael. AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.
- [Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4) :469–472, 1985.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4) :792–807, 1986.
- [KL07] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and Hall CRC Press, 2007.
- [LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2) :373–386, 1988.
- [McE87] Robert McEliece. *Finite fields for computer scientists and engineers*. Kluwer Academic Publishers, 1987.
- [Mer89] Ralph C. Merkle. A certified digital signature. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 218–238, 1989.
- [Mil76] Gary L. Miller. Riemann’s hypothesis and tests for primality. *J. Comput. Syst. Sci.*, 13(3) :300–317, 1976.
- [MMO85] S. Matyas, C. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithms. *IBM Technical Disclosure Bulletin*, 27 :5658–5659, 1985.
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2) :151–158, 1991.
- [NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2) :231–262, 2004.
- [Rab80] Michael O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comput.*, 9(2) :273–280, 1980.
- [Rog15] Phillip Rogaway. The moral character of cryptographic work, 2015.

- [RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, 1978.
- [Sha49] Claude Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4) :656–715, 1949.
- [Sho05] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, 2005.
- [Sin99] Simon Singh. *The code book*. Harper Collins, 1999.

Index

- O -notation, 29
- \mathbf{Z}_m , anneau, 80
- \mathbf{Z}_m , corps, 80
- équivalence calculatoire, 30
- AES, système, 13
- DES, système, 13
- NP, classe de complexité, 36
- P, classe de complexité, 36

- indice de coïncidence, 17

- AES, 54
- affine, système, 16
- algorithme polynomial, 29
- algorithme PPT, 29
- analyse de fréquence, 15
- anneau produit, 82
- anonymat, propriété, 12
- argument hybride, 31

- Bezout, solutions congruence, 76
- Bezout, théorème, 76

- César, système, 15
- Carmichael, nombre, 85
- carré itéré, exposant, 81
- CBC, tolérance aux erreurs, 100
- chiffrement double, 53
- compression CHP, 69
- compression, Davies-Meyer, 68
- confidentialité parfaite, 22
- confidentialité, 1 message, 34
- confidentialité, CCA, 43
- confidentialité, CPA, 41
- confidentialité, propriété, 11
- congruence, 80
- congruence quadratique, 85
- cyclique, groupe, 86

- déterminant, 89
- DES, 51
- Diffie-Hellman, en groupe, 102
- Diffie-Hellman, protocole, 62
- distance statistique, 25
- divisibilité, 75

- El Gamal, système, 13
- ElGamal, système, 63
- entiers de Blum, 103

- espace de probabilité, 22
- Euclide, algorithme, 77
- Euler, fonction, 83
- exponentiation rapide, 81

- famille v.a.d, 30
- Fermat, petit théorème, 84
- Fermat, test primalité, 85
- fonction à sens unique, 19, 36
- fonction affine, inversion, 90
- fonction de coût, 29
- fonction négligeable, 30

- générateur affine, attaque, 99
- générateur d'un groupe, 86
- générateur, recherche, 88
- Gnu Privacy Guard, 91
- groupe cyclique, 86
- groupe multiplicatif, 83

- Hill, système, 18

- intégrité, 48
- intégrité, propriété, 12
- inverse multiplicative, 80

- Kasinski, attaque, 17

- Lagrange, cardinalité sous-groupe, 84
- logarithme discret, 88
- logarithme discret, calcul, 97

- MAC, 48
- MAC, mode CBC, 48
- matrice adjointe, 89
- matrice inverse, 89
- Merkle-Damgård, construction, 66
- Miller-Rabin, test primalité, 86
- mise en gage d'un bit, 93
- mode chaîné (CBC), 44
- mode compteur (CTR), 43
- mode dictionnaire (ECB), 14
- module, 75
- mono-alphabétique, système, 15

- nombre de générateurs, 87
- non-répudiation, propriété, 12

- ordre d'un élément dans un groupe, 83

ou exclusif, 21

paradoxe des anniversaires, 66

pgcd, 76

pile ou face, protocole, 19

poly-alphabétique, système, 16

premier, nombre, 79

premiers, dénombrables, 79

premiers, distribution, 79

PRF, 40

PRF de PRG, construction GGM, 40

PRG, 33

probabilité conditionnelle, 22

PRP de PRF (Luby-Rackoff), 45

pseudo-primalité, 96

quotient et reste, 75

restes chinois, théorème, 82

routage anonyme, 95

RSA, attaque sur petit exposant, 102

RSA, système, 13, 60

schéma de Feistel, 45

Shannon, théorème, 23

signature aveugle, 97

signature RSA, 71

signature RSA à anneau, 102

substitution, système, 15

système à masque jetable, 21

système cryptographique, 12

système hybride, 59

système, asymétrique, 13

système, symétrique, 13

test pour les résidus quadratiques, 103

texte clair connu, attaque, 18

transposition, système, 18

Vigenère, système, 16