Université Bordeaux I

Master CSI

Année 2004-2005

Cours de cryptographie symétrique

Christine Bachoc

Bibliography

[1] *Sequence analysis*, Guang Gong, disponible sur le web à l'URL: http://www.cacr.math.uwaterloo.ca/ ggong

[2] *PGP*, Simon Garfinkel. OReilly (1995)

[3] *Handbook of applied cryptography*, Menezes, van Oorschot, Vanstone, disponible sur le web à l'URL: http://www.cacr.math.uwaterloo.ca/hac

[4] *Cryptographie appliquée*, Bruce Schneier. International Thomson Publishing France (1997)

[5] *Histoire des codes secrets*, Simon Singh

[6] *Cryptographie théorie et pratique*, Douglas Stinson. International Thomson Publishing France (1996)

[7] *Cours de cryptographie*, Gilles Zemor, Editions Cassini

Chapter 1

Introduction

Le chiffrement d'un message consiste à le coder (c'est le terme commun, mais en cryptographie, on dit plutôt "chiffrer" que "coder") pour le rendre incompréhensible pour quiconque n'est pas doté d'une clé de déchiffrement K_D (qui doit donc impérativement être gardée secrète):

Le chiffrement est une activité cryptographique très ancienne qui remonte à l'antiquité (6ème siècle avant J.C.).

Dans un cryptosystème, on distingue:
1 - l'espace des messages clairs $\mathcal{M}$ sur un alphabet $\mathcal{A}$ (qui peut être l'alphabet romain, mais qui sera dans la pratique $\{0, 1\}$ car tout message sera codé en binaire pour pouvoir être traité par ordinateur);
2 - l'espace des messages chiffrés $\mathcal{C}$ sur un alphabet $\mathcal{B}$ (en général égal à $\mathcal{A}$);
3 - l'espace des clés $\mathcal{K}$
4 - un ensemble $\mathcal{E}$ de transformations de chiffrement (chaque transformation étant indexée par une clé):

$$E_K : M \in \mathcal{M} \mapsto C \in \mathcal{C}$$

5 - un ensemble $\mathcal{D}$ de transformations de déchiffrement (chaque transformation étant indexée par une clé):

$$D_K : C \in \mathcal{C} \mapsto M \in \mathcal{M}.$$

Evidemment, la condition $D_{K_D}(E_{K_C}(M)) = M$ doit être réalisée.

On notera un tel cryptosystème $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$.

Rappelons ici le *principe de Kerkhoffs*, énoncé dès la fin du XIXème siècle par ce dernier: la sécurité d'un système cryptographique ne doit pas reposer sur la non divulgation des fonctions de chiffrement et de déchiffrement utilisées mais sur la non divulgation des clés utilisées pour les paramétrer.

Jusqu'au milieu des année 70, les seuls cryptosystèmes connus étaient *symétriques* (on dit aussi *conventionnels* ou *à clé secrète*): la clé de chiffrement K_C était la même que la clé de déchiffrement K_D (ou du moins, la connaissance de la clé de chiffrement permettait d'en déduire la clé de déchiffrement) ce qui obligeait à garder secrète la clé K_C elle aussi. Se pose alors le problème crucial de l'échange de clé, impossible à résoudre dans le cas d'un système développé à grande échelle.

En 1976, W. Diffie et M. Hellman introduisirent le concept de cryptographie *à clé publique* (ou *asymétrique*): dans ce type de système, la clé de chiffrement est publique, c'est à dire connue de tous. Seule la clé de déchiffrement reste secrète. Si n personnes veulent communiquer secrètement 2 à 2, il leur faut en tout $2n$ clés (chacune détient une clé secrète et diffuse une clé publique), alors que si elles utilisent un chiffrement conventionnel, il leur faut une clé secrète pour chaque paire de correspondants, c'est à dire en tout $\binom{n}{2} = \frac{n(n-1)}{2}$ clés. Surtout, dans un système asymétrique, la clé publique *par définition* peut être connue de tous, et la clé de déchiffrement n'a pas besoin d'être transmise à son correspondant.

En 1978, le premier système de chiffrement à clé publique fut introduit par R. Rivest, A. Shamir et L. Adleman: le système RSA. Ce système est un des seuls qui aient résisté à la cryptanalyse.

Mais cela n'a pas pour autant sonné la fin de la cryptographie conventionnelle car celle-ci est beaucoup plus rapide à mettre en oeuvre (entre 100 et 1000 fois plus rapide selon les cryptosystèmes) et ses clés sont plus courtes (un peu plus d'une centaine de bits au lieu d'au moins un millier) pour un même niveau de sécurité. Dans la pratique (voir par exemple PGP, *Pretty Good Privacy*), à moins qu'on ait à communiquer des messages de petite taille (de quelques k-octets), on agit comme suit. Supposons qu'Alice veuille envoyer un message secret à Bob.

- Alice choisit un cryptosystème à clé publique et utilise ce système pour envoyer secrètement à Bob un mot binaire K, qu'on appelle clé de session; pour ce faire, elle chiffre K au moyen de la clé publique de Bob (une alternative possible est

l'utilisation d'un protocole d'échange de clés tel que celui de Diffie-Hellman);
- Bob récupère K à l'aide de sa clé secrète;
- Alice et Bob détiennent maintenant un mot secret K et Alice l'utilise pour chiffrer son message long dans un cryptosystème symétrique;
- Bob peut déchiffrer le message à l'aide de K.

La cryptographie à clé publique a donné naissance à d'autres concepts comme celui de signature numérique (où l'authenticité de l'émetteur doit être assurée).

Dans ce cours, nous nous préoccuperons uniquement de cryptographie symétrique.

Chapter 2

Les schémas cryptographiques anciens

Dans ce chapitre, nous faisons un bref tour d'horizon des systèmes cryptographiques classiques utilisés jusqu'à la deuxième guerre mondiale. S'ils sont caduques à l'heure actuelle, ils n'en restent pas moins des exemples importants qui permettent de comprendre les principes sous-jacents aux systèmes modernes et peuvent toujours servir à épater vos petits neveux...

2.1 Chiffrement par substitution

2.1.1 Substitution monoalphabétique

$\mathcal{A}$ est l'alphabet romain et $\mathcal{B} = \mathcal{A}$. Soit π une permutation (i.e. une bijection) sur $\mathcal{A}$. L'opération de chiffrement d'un message $m = m_1 \cdots m_n$ est:

$$c = E_\pi(m) = \pi(m_1) \cdots \pi(m_n).$$

La clé secrète est π. Connaissant cette clé, on peut déchiffer:

$$D_\pi(c) = \pi^{-1}(c_1) \cdots \pi^{-1}(c_n) = m.$$

Un cas particulier est quand π est le décalage circulaire de k de lettres (c'est le nombre k qui est alors le secret). Le chiffre de César utilisait $k = 3$. Dans le cas particulier du décalage circulaire, il n'y a que 26 clés possibles. Le système n'est donc pas fiable: on peut essayer tous les décalages possibles jusqu'à obtenir un texte intelligible.

Dans le cas général, il y a 26! permutations possibles et la taille de l'espace des clés est amplement suffisant (par la formule de Stirling, on a $26! \approx \frac{1}{\sqrt{2 \cdot 3,14 \cdot 26}} 26^{26} e^{-26}$

et on peut même calculer directement sur machine $26! = 403291461126605635584000000 \approx 4 \cdot 10^{26}$). Pourtant, ce système est cassé par *la cryptanalyse statistique* car dans une langue usuelle telle que le français, les lettres n'apparaissent pas toutes avec les mêmes fréquences. On repère par exemple le "e" comme la lettre la plus fréquente. D'autres particularités (par exemple, le "q" est très souvent suivi du "u") peuvent être utilisées et entrées dans un programme informatique qui retrouve la clé π immédiatement. A la simple lecture du texte chiffré, on retrouve ainsi le clair. Il s'agit d'une *attaque à chiffré seul*.

2.1.2 Substitution polyalphabétique

Au lieu d'une permutation, on dispose de t permutations $\pi_1, \cdots, \pi_t$. On applique π_1 à m_1, π_2 à m_2, ..., π_t à m_t puis on continue cycliquement en appliquant π_1 à m_{t+1}, π_2 à m_{t+2} etc... En d'autres termes:

$$c = E_{\pi_1,\cdots,\pi_t}(m) = \pi_1(m_1) \cdots \pi_{i[t]}(m_i) \cdots \pi_{n[t]}(m_n)$$

où $i[t]$ désigne i mod t si i mod $t \neq 0$ et t si i mod $t = 0$.

Par exemple, dans le système de Vigenère (utilisé massivement jusqu'au 19ème siècle), les permutations π_i étaient des décalages circulaires. Le décalage correspondant à chaque permutation était indiqué par une lettre. Ainsi, le mot "EXEMPLE" indiquerait que $t = 7$ et que π_1 est un décalage de 4 positions (car le "E" est situé à 4 positions du "A"), π_2 est un décalage de 23 positions etc...

Ce système peut s'attaquer par une méthode de type statistique.

La machine Enigma, utilisée par les allemands pendant la deuxième guerre mondiale, chiffrait par un système de Vigenère: la position initiale des rotors déterminait la suite des permutations utilisées.

2.2 Chiffrement par transposition

$\mathcal{A}$ est l'alphabet romain et $\mathcal{B} = \mathcal{A}$. Soit σ une permutation (i.e. une bijection) sur $\{1, \cdots, n\}$ où n est la longueur du clair (on peut aussi découper le clair en différents blocs et chiffrer chaque bloc). L'opération de chiffrement d'un message $m = m_1 \cdots m_n$ est:

$$c = E_\sigma(m) = m_{\sigma(1)} \cdots m_{\sigma(n)}.$$

La clé secrète est σ. Connaissant cette clé, on peut déchiffer:

$$D_\sigma(c) = c_{\sigma^{-1}(1)} \cdots c_{\sigma^{-1}(n)} = m.$$

Il y a n! permutations possibles et la taille de l'espace des clés est amplement suffisant dès que n est assez grand. Pourtant, ce système est cassé car il est *linéaire*: il existe une matrice (ici, une matrice de permutation) A_σ telle que

$$c = m \times A_\sigma.$$

Le terme $(A_\sigma)_{i,j}$ de cette matrice qui est situé à la $i$ème ligne et à la $j$ème colonne vaut 1 si $i = \sigma(j)$ et 0 sinon.

Nous décrivons maintenant l'attaque valable sur les cryptosystèmes linéaires. Soit un cryptosystème tel que $c = m \times A$ (où la matrice A dépend de la clé secrète, nous considérerons qu'elle est la clé secrète). On suppose que l'attaquant dispose de n paires de clairs-chiffrés et on va montrer qu'il peut avec une probabilité non négligeable retrouver la clé, c'est à dire retrouver A (on dit qu'il fait une *attaque à clair connu*). On met les n clairs en une matrice M carrée $n \times n$ et les n chiffrés correspondants en une matrice C. On a

$$C = M \times A.$$

Si la matrice M est inversible il en déduit

$$A = M^{-1} \times C$$

en n^3 opérations élémentaires (par la méthode classique du pivot de Gauss). La proportion de matrices inversibles parmi les matrices carrées étant non négigeable, la probabilité que l'attaque réussisse est non négigeable.

2.3 Confusion, diffusion

Les systèmes de chiffrement précédents sont des systèmes de chiffrements *par blocs*, c'est-à-dire que l'on découpe d'abord le texte clair en blocs de même taille, et le chiffrement est obtenu par l'application d'une fonction de chiffrement E_K à chacun des blocs. Dans le monde moderne, les messages sont des suites de bits (par exemple obtenus à partir de l'alphabet courant par le code ASCII). Si la taille des blocs est ℓ, supposons pour simplifier que la fonction de chiffrement E_K définisse une bijection $E_K : \mathbb{F}_2^\ell \to \mathbb{F}_2^\ell$.

Le premier exemple montre que, si ℓ est trop petit, la répartition des blocs de taille ℓ dans les messages clairs ne suivant pas une loi uniforme, le cryptosystème est sensible à une analyse statistique. Si ℓ devient grand, et si E_K est vraiment quelconque, le problème est de calculer $E_K(m)$. Si l'on doit passer par le stockage en mémoire des valeurs prises par $E_K(m)$ cela limite la taille de ℓ (justement,

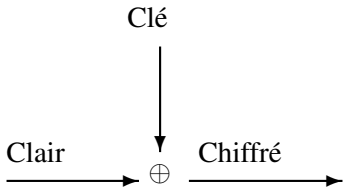
on peut prendre comme *définition* de la notion de permutation quelconque la propriété suivant laquelle il n'y a pas de méthode plus simple par exemple en terme de longueur de programme pour la définir que de donner toutes ses images). Si à l'opposé on choisit une fonction extrêmement facile à stocker et à calculer, on tombe sur un système du type linéaire comme dans le deuxième exemple.

Très schématiquement on dit qu'un système avec ℓ petit et E_K quelconque apporte de la *confusion* tandis qu'un système linéaire où ℓ est choisit grand apporte de la *diffusion* (i.e. un bit donné du message clair influence un grand nombre de bits du message chiffré).

Les systèmes par blocs modernes, comme on le verra plus précisément, alternent les séquences de confusion et de diffusion.

2.4 Le chiffrement de Vernam

Ce système (aussi appelé masque jetable ou one-time-pad) date de 1917 et a servi à chiffrer les message télégraphiques.



Les messages, clairs et chiffrés, sont des suites de bits de longueur n (i.e. appartiennent à $\mathbb{F}_2^n$). La clé K est une suite binaire de la même longueur que le clair. Le chiffrement consiste en l'ajout (mod 2) bit à bit du clair et de la clé:

$$c = m \oplus K = m_1 \oplus K_1 \cdots m_n \oplus K_n.$$

Ce système est parfait dans le cadre d'une attaque à chiffré seul, en le sens que si l'attaquant peut décrypter un (seul) message, c'est-à-dire trouver m, connaissant c, alors c'est qu'il détient la clé. En effet, celle-ci se déduit de m et de c par:

$$K = m \oplus c.$$

Mais, pour la même raison, il est vulnérable aux attaques à clairs connus, ce qui implique que la clé ne doit être utilisée qu'une fois!

De plus, il est impratiquable, sauf pour des messages de petites longueurs: la clé devant être envoyée sur un canal sécurisé (le système est symétrique!), devant être aussi longue que le message et ne devant être utilisée qu'une fois, il semble aussi simple d'envoyer le message lui-même sur le canal sécurisé. Cependant, ce système a été utilisé pour le téléphone rouge car il est préférable d'envoyer la clé (par porteur dans le cas du téléphone rouge) et d'en renvoyer une autre si elle a été interceptée plutôt que d'envoyer directement le message (que faire s'il est intercepté?). De plus, ce système est la base de la cryptographie par flots, qui concerne tous les chiffrements en ligne (i.e. très rapides), qui sont utilisés en particulier par les armées, pour la téléphonie mobile, etc..

Chapter 3

La théorie de Shannon

Dès 1949, Claude Shannon tente de donner des fondements théoriques à la cryptologie. Il adopte le point de vue de la théorie de l'information et introduit la notion de *sécurité inconditionnelle*.

3.1 L'entropie d'une variable aléatoire

Si on considère les éléments d'un système cryptographique: message clair, chiffré, clé, comme des variables aléatoires, on peut mesurer la quantité d'information qu'ils contiennent par leur *entropie*.

Si X est une variable aléatoire prenant un nombre fini de valeurs, et si par exemple $P(X = x) = 1/4$, on peut considérer que l'information contenue dans l'évènement $X = x$ est de 2 bits, puisque il y a exactement 4 mots à 2 bits. Plus généralement, on considère que l'information contenue dans l'évènement $X = x$ est de $-\log P(X = x)$ où $\log$ est le log en base 2 et si l'entropie de X mesure l'information moyenne contenue dans X, il est naturel de poser:

Définition 1 *L'entropie $H(X)$ de la variable aléatoire X est par définition:*

$$H(X) := -\sum_x P(X = x) \log P(X = x).$$

À cause de la concavité de la fonction $x \to -x \log x$ sur l'intervalle $[0, 1]$, l'entropie $H(X)$ est maximale si et seulement si X est équidistribuée, auquel cas $H(X) = \log n$ où n est le nombre de valeurs que prend X.

L'*entropie conditionnelle* de X sachant Y mesure l'incertitude résiduelle associée à X connaissant Y.

Définition 2 *L'entropie conditionnelle de X sachant Y est définie par:*

$$H(X \mid Y) := -\sum_{x,y} P(X = x, Y = y) \log P(X = x \mid Y = y).$$

Proposition 1 *On a: $H(X,Y) = H(Y) + H(X \mid Y)$ ($H(X,Y)$ désigne l'entropie de la variable aléatoire (X,Y)).*

Preuve: Il suffit de se rappeler que

$$P(X = x, Y = y) = P(X = x \mid Y = y)P(Y = y).$$

On a alors:

$$\begin{aligned}
H(X,Y) &= -\sum_{x,y} P(X = x, Y = y) \log P(X = x, Y = y) \\
&= -\sum_{x,y} P(X = x, Y = y)(\log P(X = x \mid Y = y) + \log P(Y = y)) \\
&= H(X \mid Y) - \sum_{y}(\sum_{x} P(X = x, Y = y)) \log P(Y = y) \\
&= H(X \mid Y) - \sum_{y} P(Y = y) \log P(Y = y) \\
&= H(X \mid Y) + H(Y)
\end{aligned}$$

3.2 Sécurité inconditionnelle

Considérons les messages clairs, chiffrés et les clés comme des variables aléatoires M, C, K. On suppose que chaque clé k définit une bijection $e_k : \mathcal{M} \to \mathcal{C}$. On a:

$$P(C = c) = \sum_{k} P(K = k) P(M = e_k^{-1}(c)).$$

Shannon définit un cryptosystème comme *parfait*, ou assurant une *sécurité inconditionnelle*, si la condition:

$$\boxed{H(M \mid C) = H(M)}$$

est réalisée. Autrement dit, la connaissance du message chiffré n'apporte aucune information sur le message clair.

Exemple: Le chiffrement de Vernam est parfait, si K est une variable équidistribuée, et indépendante de M. En effet,

$$P(M = x, C = y) = P(M = x, K = y - x) = \frac{P(M = x)}{|\mathcal{K}|}$$

en utilisant l'indépendance de K et M et l'équidistribution de K. On en déduit également:

$$P(C = y) = \sum_x P(M = x, C = y) = \frac{1}{|\mathcal{K}|}.$$

Donc:

$$P(M = x \mid C = y) = \frac{P(M = x, C = y)}{P(C = y)} = P(M = x)$$

et on a bien $H(M \mid C) = H(M)$.

En général, on voit facilement que:

$$\begin{aligned} H(M \mid C) &\leq H((M, K) \mid C) \\ &= H(K \mid C) \\ &\leq H(K) \end{aligned}$$

ce qui démontre l'un des résultats fondamentaux de la théorie de Shannon:

Théorème 1 *Si le système cryptographique est parfait, on a :*

$$H(K) \geq H(M).$$

L'information contenue dans la clé est donc au moins aussi grande que celle du message clair. En particulier, si tous les messages clairs sont équiprobables, la taille de la clé doit être au moins aussi grande que celle du message clair. On ne peut donc faire mieux que le chiffrement de Vernam!

Puisqu'il faut quand même faire quelque chose de plus raisonnable dans la pratique, on remplace maintenant la notion de sécurité inconditionnelle par celle de sécurité calculatoire. En effet, dans le modèle de Shannon, l'attaquant a des ressources calculatoires illimitées ce qui n'est certainement pas le cas de tout un chacun...

Chapter 4

La cryptographie symétrique moderne: généralités

4.1 Les différents types d'attaques

L'objectif de l'attaquant c'est de trouver la clé. C'est plus fort que de trouver le clair associé à un chiffré donné, puisque la donnée de la clé fournit la possibilité de calculer tous les clairs associés à tous les chiffrés. On considère qu'une attaque est efficace si elle a une probabilité non négigeable de réussir en un temps inférieur ou égal à quelques années (voire plus!) sur une ou plusieurs machines puissantes. Cela fixe de nos jours à 2^{80} le nombre minimal d'opérations élémentaires nécessaires à une attaque, pour qu'un cryptosystème soit considéré comme sûr.

Il a à sa disposition différents types d'attaques possibles. La plus évidente est

- *l'attaque exhaustive*. L'attaquant essai toutes des clés k jusqu'à ce que $D_k(c)$ ressemble à un texte clair (ce qui est d'ailleurs un concept à préciser, mais auquel on peut donner un sens statistique à peu près intuitif). Le temps moyen de cette attaque est égal au temps d'un déchiffrement multiplié par la moitié de la taille de l'espace des clés (exercice!).

La parade bien sûr consiste à choisir un système crypto dont l'espace des clés a une taille suffisante. Le DES, premier standard de chiffrement symétrique, a été écarté précisément parce que la taille des clés utilisées est insuffisante: 56 bits.

Ensuite on a, par ordre décroissant de difficulté pour l'attaquant:

- *l'attaque à chiffré seul* (vue dans la section sur le chiffrement par substitution): l'attaquant ne connait que le chiffré et il cherche le clair et si possible la clé;
- *l'attaque à clair connu* (vue dans la section sur le chiffrement par transposition, qui est linéaire): l'attaquant connait un ou plusieurs (éventuellement un grand nombre) de couples clairs-chiffrés, et il cherche la clé;

- *l'attaque à clair choisi*, semblable à la précédente, mais pour laquelle l'attaquant peut choisir les clairs de façon que son attaque réussisse avec une meilleure probabilité (elle peut être *adaptative* si les choix peuvent être faits en fonction des résultats des attaques sur les couples précédents);
- *l'attaque à chiffré choisi*, qui peut être elle aussi *adaptative*.

Le niveau normal de sécurité est la résistance à l'attaque à clair choisi. Un bon système est tel que cette attaque ait une complexité (en temps de calcul) comparable à celle de l'attaque exhaustive.

4.2 Chiffrement par flot, par blocs

Les schémas de chiffrement par flot, traitent l'information bit à bit, et sont très rapides. Ils peuvent être traités avec une mémoire limitée et la propagation des erreurs de transmission du chiffré au clair est limitée. Ils sont parfaitement adaptés à des moyens de calcul, de mémoire et de transmission limités (cryptographie en temps réel) comme la cryptographie militaire, ou la cryptographie entre le téléphone portable GSM et son réseau (système A51).

Leur principe est d'effectuer un chiffrement de Vernam en utilisant une clé pseudo-aléatoire, c'est à dire une clé qui ne soit pas choisie aléatoirement parmi tous les mots binaires de longueur n. Cette clé (qu'on appellera *suite pseudo-aléatoire*) est générée par différents procédés à partir d'une clé aléatoire d'une longueur juste suffisante pour résister aux attaques exhaustives (les performances à venir des ordinateurs demandent donc de la prendre d'au moins 80 bits), qu'on appellera *clé courte*.

La sécurité d'un tel système contre une attaque à clair connu est équivalente à la sécurité du générateur pseudo-aléatoire. On demande bien entendu que la connaissance de N bits consécutifs de la clé ne permette pas de calculer facilement la clé courte. On est plus exigeant en demandant que *la connaissance de N bits consécutifs ne permette pas de prévoir facilement (i.e. en temps polynomial en N) la valeur du bit suivant avec probabilité $p > 1/2$*. En particulier cela demande que la suite des bits de la clé satisfasse aux propriétés de régularité d'une suite véritablement aléatoire.

Les schémas par blocs sont plus lents et nécessitent plus de moyens informatiques que les schémas par flots. Mais ils sont bien adaptés à la cryptographie civile comme celle des banques.

Dans un système par blocs, chaque clair est découpé en blocs de même longueur et chiffré bloc par bloc. La longueur l des clés doit être suffisante pour que l'attaque exhaustive (attaque à chiffré seul si on a un moyen de différencier les clairs des

messages aléatoires, et attaque à clair connu sinon) consistant à déchiffrer le chiffré avec toutes les clés possibles jusqu'à l'obtention du clair, soit irréaliste ($l \geq 80$). La longueur n des blocs doit également être suffisante pour éviter les attaques dites par dictionnaire consistant à s'aider d'un pré-calcul (partiel) des chiffrés des 2^n blocs possibles.

La sécurité des schémas par blocs retenus comme standards (DES puis AES) repose sur le fait qu'avant d'être retenus (et après!), ils ont été massivement attaqués par la communauté cryptographique. Ces schémas qui ont résisté sont alors considérés comme sûrs. Le DES (Data Encryption Standard) date des années 70 et a résisté à toutes les techniques de cryptanalyse (l'attaque qui reste la plus efficace en pratique est l'attaque exhaustive). Il a dû cependant être remplacé récemment comme standard par l'AES (Advanced Encryption Standard), car sa clé de 56 bits était trop courte (et de longueur non modifiable!) pour assurer de nos jours une résistance suffisante à l'attaque exhaustive.

Les exemples historiques de chiffrement (par transposition et par substitution) vus en début de cours sont des chiffrements par blocs. La substitution ajoute de la *confusion* au procédé de chiffrement et la transposition ajoute de la *diffusion* en éparpillant l'influence moyenne (selon les différentes clés) de chaque bit du clair, sur les bits du chiffré. Mais aucun de ces deux procédés ne produit à la fois de la confusion et de la diffusion, et c'est une raison pour laquelle ils ne peuvent pas à assurer une réelle sécurité. Les systèmes modernes, pour assurer une véritable sécurité, doivent produire à la fois de la confusion et de la diffusion, faute de quoi ils ne résistent pas aux attaques que nous décrirons plus loin.

Définition 3 *On appelle chiffrement produit un chiffrement par blocs qui combine plusieurs transformations élémentaires (substitutions, transpositions, opérations linéaires ou arithmétiques).*
Un chiffrement itératif résulte de l'application itérée d'un chiffrement (en général un chiffrement produit). Chaque itération est appelée un tour (round en anglais). Chaque tour fait intervenir une sous-clé qui en général est dérivée (on dit souvent cadencée) de la clé principale.

Chapter 5

Les schémas de chiffrement par flot

Pour les notions de base sur l'analyse de séquences, on pourra consulter utilement [1].

5.1 Critères statistiques

On demande à une suite pseudo-aléatoire, périodique de période N, de satisfaire certaines des propriétés des suites authentiquement aléatoires. Parmi ces propriétés, les plus classiques sont les trois **critères de Golomb**, proposés par celui-ci en 1982:

1. Dans chaque période, le nombre de 0 est approximativement égal au nombre de 1: $|\sum_{i=0}^{N-1}(-1)^{s_i}| \leq 1$.

2. Une série (de 0 ou de 1) est une succession de bits identiques, maximale (i.e. encadrée par des bits opposés). Dans chaque période, soit S l'ensemble des séries; si $2^k \leq |S| < 2^{k+1}$, on trouve $|S|/2$ séries de longueur 1, $|S|/4$ séries de longueur 2, ..., $|S|/2^k$ séries de longueur k, et pour chaque longueur, autant de séries de 0 que de séries de 1.

3. La fonction d'auto-corrélation prend deux valeurs, suivant que $\tau = 0$ ou non, ce qui signifie que s est indépendante de ses translatées.

$$C(\tau) := \sum_{i=0}^{N-1}(-1)^{s_i+s_{i+\tau}}.$$

Bien sûr, ces conditions ne garantissent en aucun cas la sécurité cryptographique! On va voir au prochain paragraphe un procédé facile pour engendrer des suites pseudo-aléatoires satisfaisant ces critéres. D'autres critères sont usuellement retenus, notamment la notion de *profil de complexité*, expliquée plus loin.

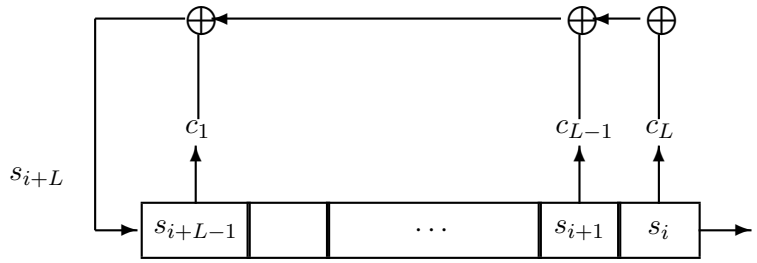
5.2 Les registres à décalage linéaires (LFSR).

Un premier procédé consiste à prendre pour clé pseudo-aléatoire une suite (on dit aussi séquence) à récurrence linéaire, c'est à dire une suite binaire satisfaisant une relation de récurrence du type:

$$s_t = \sum_{i=1}^{L} c_i s_{t-i}, \forall t \geq L, \tag{5.1}$$

où les c_j sont non tous nuls, et la somme est l'addition dans $\mathbb{F}_2$.

Le procédé électronique pour générer de telles suites est le registre à décalage linéaire (*Linear Feedback Shift Register* en anglais; on écrira en abrégé LFSR). Un registre à décalage linéaire est constitué de L cases mémoires appelées flip-



flops pouvant chacune contenir un bit, et d'une horloge contrôlant le mouvement des données (ce mouvement a lieu de la gauche vers la droite dans le schéma ci-joint). Le registre est initialisé par L bits $s_0, \cdots, s_{L-1}$ placés de droite à gauche. A chaque top d'horloge, sort le bit situé le plus à droite juste avant le top (les L premiers bits à sortir sont donc bien les bits $s_0, \cdots, s_{L-1}$ dans cet ordre). Les autres bits se déplacent d'un rang vers la droite et le flip-flop le plus à gauche reçoit comme nouvelle valeur la combinaison linéaire

$$s_{L+i} = \sum_{j=1}^{L} c_j s_{L+i-j}.$$

La suite générée satisfait donc bien cette relation de récurrence à partir du rang L.

Notons que si la suite initiale $s_0, \cdots, s_{L-1}$ est nulle, alors toute la suite est nulle. La suite pseudo-aléatoire est constituée des bits de sortie successifs du LFSR pendant un certain nombre de tops d'horloge et la clé courte est constituée de l'initialisation du registre et des coefficients c_i eux-mêmes ($2L$ bits en tout). Les coefficients c_i sont en effet supposés secrets: s'ils sont connus, alors en observant L bits consécutifs, on peut calculer tous les suivants.

Rappelons ce résultat d'algèbre linéaire élémentaire:

Proposition 2 *Supposons fixés $(c_1, c_2, \ldots, c_L)$. Soit $s = (s_n)_{n\geq 0}$ une suite binaire vérifiant la relation de récurrence (5.1). La donnée des L premières valeurs $(s_0, s_1, \ldots, s_{L-1})$ détermine de façon unique la suite $s = (s_n)_{n\geq 0}$. Plus précisément, l'application qui à la suite s associe ses L premiers termes est un isomorphisme de $\mathbb{F}_2$-espaces vectoriels de l'espace des suites récurrentes pour les coefficients $(c_1, c_2, \ldots, c_L)$ sur $\mathbb{F}_2^L$.*

En particulier, il y a exactement 2^L suites différentes, satisfaisant la même relation de récurrence, associées aux 2^L possibilités pour l'initialisation $(s_0, s_1, \ldots, s_{L-1})$.

Nous allons voir maintenant qu'une telle suite est en fait périodique.

Proposition 3 *La suite s est ultimement périodique, de période $T \leq 2^L - 1$ (i.e. il existe un entier i_0 tel que $s_i = s_{i+T}$ pour tout $i \geq i_0$). Si, de plus, $c_L = 1$, la suite s est périodique (i.e. $s_i = s_{i+T}$ pour tout $i \geq 0$).*

Preuve: Notons $R_i := (s_i, s_{i+1}, \ldots, s_{i+L-1})$ le i-ème registre. Celui-ci détermine complètement les registres ultérieurs. Ce registre peut prendre au plus 2^L états. S'il atteint l'état $0 = (0, \ldots, 0)$ alors les registres successifs sont tous nuls et la suite elle-même est nulle à partir de là. S'il n'est jamais nul, parmi $[R_0, R_1, \ldots, R_{2^L-1}]$, au moins deux registres sont identiques. Supposons $R_{i_0} = R_{i_0+T}$; alors la suite des registres $[R_{i_0}, R_{i_0+1}, \ldots, R_{i_0+T-1}]$ se répète indéfiniment. On a donc $s_i = s_{i+T}$ pour tout $i \geq i_0$ avec $T \leq 2^L - 1$.

On peut interpréter aussi la relation entre deux registres successifs en termes matriciels: Notons

$$A := \begin{pmatrix} 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 1 & 0 \\ 0 & 0 & \ldots & 0 & 0 & 1 \\ c_L & c_{L-1} & \ldots & c_3 & c_2 & c_1 \end{pmatrix}$$

En considérant les R_i comme des vecteurs colonnes, on a

$$R_{i+1} = AR_i.$$

Ainsi, on a $R_n = A^n R_0$. Remarquons que le déterminant de A est égal à c_0. Ainsi, si $c_0 = 1$, la matrice A est inversible et le LFSR ne passe jamais par le registre nul. La condition $R_{i_0} = R_{i_0+T}$ devient $A^{i_0} R_0 = A^{i_0+T} R_0$ mais comme A est inversible, on en déduit $R_0 = A^T R_0 = R_T$ et la suite s est périodique.

Remarquons que, si $c_0 = 0$, en notant i_0 le plus petit indice tel que $c_{i_0} \neq 0$, la suite s peut se voir comme une suite avec un préfixe $s_0, s_1, \ldots, s_{i_0-1}$, suivie d'une suite engendrée par un LFSR de longueur $L - i_0$, qui lui vérifie $c'_0 = c_{i_0} \neq 0$.

□

On supposera désormais que la suite s est périodique, et non nulle. Clairement, pour des applications cryptographiques, il est souhaitable que la suite s ait une période aussi longue que possible, c'est-à-dire, d'après ce qui précède, une période $T = 2^L - 1$. Remarquons que, dans ce cas, les registres R_i prennent tous les états possibles de $\mathbb{F}_2^L$ sauf 0. En particulier, deux telles suites associées aux mêmes coefficients de récurrence mais pas au même état initial sont en fait décalées l'une de l'autre.

Définition 4 *Si la suite s (avec $c_L = 1$) a pour période $2^L - 1$, on dit que s est une* m-suite *ou* m-séquence*, ou encore qu'elle est* de longueur maximale*.*

On va voir que les m-suites se caractérisent aisément en termes de propriétés de leur *polynôme de rétroaction*.

Définition 5 *Le polynôme de connexion, ou polynôme caractéristique, ou polynôme générateur de la suite s est le polynôme à coefficients dans $\mathbb{F}_2$:*

$$f(X) = 1 + c_1 X + \cdots + c_L X^L.$$

Le polynôme de rétroaction de la suite s est le polynôme réciproque du polynôme de connexion:

$$h(X) = X^L + c_1 X^{L-1} + \cdots + c_L.$$

Proposition 4
1. *Le polynôme de rétroaction est le polynôme caractéristique de la matrice A.*
2. *Si D désigne l'opérateur "décalage" sur les suites, défini par:*

$$(Ds)_i = s_{i+1} \text{ pour tout } i \geq 0,$$

on a $h(D)s = 0$. Tout polynôme h' tel que $h'(D)s = 0$, définit un LFSR engendrant s. L'ensemble de ces polynômes, est un idéal de $\mathbb{F}_2[X]$, engendré par un polynôme de plus petit degré, définissant le plus petit LFSR engendrant s.

Preuve:

1. C'est un calcul standard.

2. Si $s = s_0 s_1 s_2 \ldots$, $Ds = s_1 s_2 s_3 \ldots$, et $D^i s = s_i s_{i+1} s_{i+2} \ldots$. On a clairement: $(D^L s)_i = s_{L+i}$ et

$$(h(D)s)_i = s_{L+i} - \sum_{j=1}^{L} c_j s_{L+i-j}.$$

   La condition: $h(D)s = 0$ est donc équivalente à la condition (5.1).

   Il est clair que l'ensemble de ces polynômes est un idéal de $\mathbb{F}_2[X]$; il est donc principal, engendré par son élément non nul de plus petit degré.

□

Théorème 2 *La suite s est une m-séquence si et seulement si son polynôme de rétroaction est le polynôme minimal d'un générateur du groupe (cyclique) $\mathbb{F}_{2^L}^*$ (on dit que c'est un polynôme* **primitif***).*

Preuve: Notons $I := \{p(x) \in \mathbb{F}_2[x] \mid p(D)s = 0\}$ l'idéal défini précédemment. On sait que $h \in I$. Remarquons également que s est périodique de période T si et seulement si $x^T - 1 \in I$.

Supposons que h soit le polynôme minimal d'un générateur de $\mathbb{F}_{2^L}^*$. Comme h est irréductible, on a $I =< h >$. Ses racines (conjuguées) sont toutes d'ordre exactement $2^L - 1$. Donc h divise $x^{2^L-1} - 1$ mais ne divise pas $x^T - 1$ pour $T < 2^L - 1$. Donc s est bien de période $2^L - 1$.

Réciproquement, supposons que la suite s soit une m-suite. Notons p un générateur de I. Alors p divise $x^{2^L-1}-1$ mais ne divise pas x^T-1 pour $T < 2^L-1$. Les racines de p appartiennent donc à $\mathbb{F}_{2^L}$. Comme $p \in \mathbb{F}_2[x]$, si un élément $\alpha \in \mathbb{F}_{2^L}$ est racine de p, tous ses conjugués le sont aussi. En particulier, si l'une de ses racines est de degré L sur $\mathbb{F}_2$, c'est terminé car, h étant de degré au plus L (puisque p divise h), il ne peut être qu' irréductible et de degré L, donc égal à h. Ses racines sont toutes conjuguées, elles ont donc toutes même ordre multiplicatif, qui ne peut être que $2^L - 1$ sinon p diviserait un $x^T - 1$

pour $T < 2^L - 1$ et s ne serait pas de période maximale. Il reste à exclure le cas où toutes les racines $\alpha_1, \ldots, \alpha_s$ de p seraient de degré un diviseur strict de L. À chacune de leurs orbites sous l'action du Frobenius correspond un degré m_i et un ordre multiplicatif d_i. On a $\sum m_i = \deg(p) \leq L$. Pour que s soit de période maximale, il faudrait que $\mathrm{ppcm}(d_i) = 2^L - 1$. Mais d_i est un diviseur de $2^{m_i} - 1$. La suite d'inégalités suivante montre que c'est impossible: $\mathrm{ppcm}(d_i) \leq \mathrm{ppcm}(2^{m_i} - 1) \leq \prod(2^{m_i} - 1) < 2^{\sum m_i} - 1 \leq 2^L - 1$.

□

On a vu qu'une suite périodique est engendrée par une infinité de LFSR, associés aux éléments de l'idéal I. Bien sûr, le plus petit d'entre eux est le plus intéressant. Cela nous amène à la définition suivante:

Définition 6 *La* complexité linéaire *d'une suite périodique s est la longueur du plus petit LFSR qui l'engendre. On la note $\mathcal{L}(s)$.*

Proposition 5 *Si s est une m-suite de période $2^L - 1$, alors sa complexité linéaire est égale à L.*

Preuve: Si sa complexité etait $L' < L$, sa période serait au plus égale à $2^{L'} - 1$.

□

Résumé: Un LFSR de polynôme irréductible semble être un bon candidat pour un générateur pseudo-aléatoire. En effet, les suites non nulles qu'il engendre ont une période grande, et une complexité linéaire bien égale à L. Elles ont de bonnes propriétés statistiques: on peut montrer qu'elles vérifient les 3 critères de Golomb. De plus, on a vu que chaque L-uplet de nombres binaires apparait comme un registre une et une seule fois par période, et on peut montrer que la fréquence des mots de longueur $k \leq L$ est aussi uniforme.

Malheureusement un tel générateur n'est pas cryptographiquement sûr, à cause de l'algorithme de Berlekamp-Massey, discuté au paragraphe suivant, qui calcule (en temps quadratique) la complexité linéaire et un polynôme engendrant une suite finie. Il suit du Lemme suivant qu'il suffit de connaitre $2L$ bits consécutifs de la suite pour la retrouver entièrement.

Lemme 1 *Soit s une suite récurrente de longueur maximale, de complexité linéaire L. Si on connait $2L$ termes consécutifs de cette suite, alors on peut calculer les coefficients de récurrence en inversant un système linéaire de taille $L \times L$.*

Preuve: On a la relation linéaire:

$$\begin{pmatrix} s_i & s_{i+1} & s_{i+2} & \dots & s_{i+L-1} \\ s_{i+1} & s_{i+2} & s_{i+3} & \dots & s_{i+L} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_{i+L-2} & \vdots & \dots & \dots & s_{i+2L-3} \\ s_{i+L-1} & s_{i+L} & \dots & \dots & s_{i+2L-2} \end{pmatrix} \begin{pmatrix} c_L \\ c_{L-1} \\ \vdots \\ c_2 \\ c_1 \end{pmatrix} = \begin{pmatrix} s_{i+L} \\ s_{i+L+1} \\ \vdots \\ s_{i+2L-2} \\ s_{i+2L-1} \end{pmatrix}.$$

Il suffit de montrer que la matrice S de ce système est inversible. Ses colonnes sont les vecteurs associés aux registres $R_i, \dots, R_{i+L-1}$. Une combinaison linéaire nulle non triviale équivaut à l'existence d'un polynôme P non nul de degré au plus égal à L tel que $R_i \in \ker P(A)$. Par hypothèse, le polynôme caractéristique de A est irréductible de degré L, donc premier à P, donc $P(A)$ est inversible.

□

5.3 L'algorithme de Berlekamp-Massey

[Ce qui suit est valable sur n'importe quel corps de base. L'algorithme est une adaptation due a Massey (69) de l'algorithme de Berlekamp destiné au décodage des codes cycliques. Référence: l'article de Blahut dans le HBCT].

Dans la pratique cryptographique, l'attaquant connait seulement un bout de la suite s, et il essaie de deviner les bits suivants. En particulier, il peut chercher si ce bout de suite est le début d'un LFSR. Dans cette direction , on étend la notion de complexité linéaire aux suites finies:

Définition 7 *Étant donnée une suite $s = (s_0, \dots, s_{n-1})$, et un polynôme $f = 1 + c_1x + \dots + c_Lx^L$ (maintenant f n'est plus nécessairement de degré exactement égal à L), on dit que (L, f) engendre s si $s_j = \sum_{i=1}^{L} c_i s_{j-i}$ pour tout $j = L, \dots, n-1$. Le plus petit L convenable s'appelle la complexité linéaire de s et se note $\mathcal{L}(s)$.*

L'algorithme de Belekamp-Massey permet de calculer un (L, f) associé à s, avec $L = \mathcal{L}(s)$. En fait l'algorithme calcule un couple (L, f) avec L minimal, successivement pour les suites tronquées $s^{(1)}, s^{(2)}, \dots, s^{(k)}, \dots, s = s^{(n)}$, où $s^{(i)} = (s_0, \dots, s_{i-1})$. La suite des entiers L s'appelle aussi le *profil de complexité* de la suite s. Le temps de calcul de cet algorithme est en n^2.

On commence par quelques résultats intermédiaires.

Lemme 2 *Si (L, f) et (L', f') engendrent $s = s^{(k)}$, et si $k \geq L + L'$, alors $\sigma_k = \sigma'_k$, où $\sigma_k := \sum_{i=1}^{L} c_i s_{k-i}$ et $\sigma'_k := \sum_{i=1}^{L'} c'_i s_{k-i}$.*

Preuve: Soit $\sigma_k = \sum_{i=1}^{L} c_i s_{k-i}$. Les termes s_{k-i} peuvent s'exprimer avec f', à condition que $L' \leq k-i \leq k-1$. C'est le cas pour tout $i \leq L$ puisque $L' \leq k - L \leq k - i$ par hypothèse. Plus précisément,

$$s_{k-i} = \sum_{j=1}^{L'} c'_j s_{k-i-j}$$

et donc

$$\begin{aligned}
\sigma_k &= \sum_{i=1}^{L} c_i \sum_{j=1}^{L'} c'_j s_{k-i-j} \\
&= \sum_{j=1}^{L'} c'_j (\sum_{i=1}^{L} c_i s_{k-i-j}) \\
&= \sum_{j=1}^{L'} c'_j s_{k-j} = \sigma'_k.
\end{aligned}$$

□

Lemme 3 *Si (L, f) engendre $s^{(k)}$ mais pas $s = s^{(k+1)}$, alors $\mathcal{L}(s) \geq k - L$.*

Preuve: Supposons que (L', f') engendre s avec $L' = \mathcal{L}(s)$. Alors (L, f) et (L', f') engendrent $s^{(k)}$ et, avec les notations du Lemme 2, $s_k \neq \sigma_k$ et $s_k = \sigma'_k$ donc le Lemme 2 entraine $k < L + L'$.

□

Lemme 4 *Si (L, f) engendre $s^{(k)}$ mais pas $s^{(k+1)}$, si (M, g) engendre $s^{(m)}$ mais pas $s^{(m+1)}$ avec $m < k$, alors $(\max(L, M+k-m), f+x^{k-m}g)$ engendre $s^{(k+1)}$.*

Preuve: A la suite s on associe un polynôme $s(x) := s_0 + s_1 x + \cdots + s_{n-1}x^{n-1}$. La condition "$(L, f)$ engendre $s^{(k)}$ mais pas $s^{(k+1)}$" se traduit par: il existe des polynômes p_1 et p_2 avec $\deg(p_i) < L$ tels que:

$$s^{(k+1)}(x)f(x) = p_1(x) + x^k + x^{k+1}p_2(x).$$

De même, il existe des polynômes q_1 et q_2 avec $\deg(q_i) < M$ tels que:

$$s^{(m+1)}(x)g(x) = q_1(x) + x^m + x^{m+1}q_2(x).$$

En écrivant $s^{(k+1)}(x) = s^{(m+1)}(x) + x^{m+1}(\dots)$, on obtient

$$s^{(k+1)}(x)g(x) = q_1(x) + x^m + x^{m+1}q_3(x)$$

d'où:

$$s^{(k+1)}(x)(f(x) - x^{k-m}g(x)) = (p_1 - x^{k-m}q_1) + x^{k+1}(\dots).$$

Le degré de $p_1 - x^{k-m}q_1$ est strictement plus petit que $\max(L, M+k-m)$ et le degré de $f(x) - x^{k-m}g(x)$ est au plus égal à $\max(L, M+k-m)$. Cette dernière égalité traduit bien le fait que $(\max(L, M+k-m), f + x^{k-m}g)$ engendre $s^{(k+1)}$.

□

On peut maintenant décrire l'algorithme et sa preuve (voir [3] pour un algo prêt à implémenter).

Théorème 3 *Supposons pour $k \geq 1$ avoir calculé un couple (L, f) associé à $s^{(k)}$ avec $L = \mathcal{L}(s^{(k)})$. Soit $d_k := s_k + \sum_{i=0}^{L-1} c_i s_{k-L+i}$.*

- *Si $d_k = 0$, $\mathcal{L}(s^{(k+1)}) = L$ et (L, f) engendre $s^{(k+1)}$.*
- *Si $d_k = 1$, soit m le plus grand entier tel que $m < k$ et $\mathcal{L}(s^{(m)}) < L$, et supposons que $(\mathcal{L}(s^{(m)}), g)$ engendre $s^{(m)}$. Alors*
  - *$\mathcal{L}(s^{(k+1)}) = \max(L, k+1-L)$*
  - *$s^{(k+1)}$ est engendré par $f + x^{k-m}g$.*

Preuve: Si $d_k = 0$, c'est clair. Supposons $d_k = 1$. Notons $M = \mathcal{L}(s^{(m)})$ avec (M, g) engendrant $s^{(m)}$. L'hypothèse faite sur m (le plus grand entier tel que..) entraîne que (M, g) engendre $s^{(m)}$ mais pas $s^{(m+1)}$. L'hypothèse $d_k = 1$ entraîne que (L, f) engendre $s^{(k)}$ mais pas $s^{(k+1)}$. Le Lemme 4 montre que $s^{(k+1)}$ est engendré par $(\max(L, M+k-m), f + x^{k-m}g)$. Il reste à montrer que

$$\mathcal{L}(s^{(k+1)}) = \max(L, M+k-m) = \max(L, k+1-L).$$

On a $L \geq m+1-M$ par le Lemme 3.

Si $M+k-m \leq L$, alors, comme $k+1-L \leq M+k-m$, $L = \max(L, M+k-m), \max(L, k+1-L)$ donc c'est juste.

Si $M+k-m > L$, alors: $M+k-m \geq \mathcal{L}(s^{(k+1)}) \geq 1+k-L$, la première inégalité découlant du Lemme 4 et la deuxième du Lemme 3. Il suffit alors d'avoir $M+k-m = 1+k-L$, soit $L+M = m+1$. Cette dernière égalité se déduit du résultat suivant, que l'on démontre par récurrence: si $\mathcal{L}(s^{(m)}) \neq \mathcal{L}(s^{(m+1)})$, alors

$\mathcal{L}(s^{(m)}) + \mathcal{L}(s^{(m+1)}) = m + 1$. En effet, si cette identité est vraie pour le i-ème "saut", le raisonnement en cours montre que on l'obtient pour le $(i+1)$-ième (où forcément $d = 1$ et on n'est pas dans le cas d'égalité).

□

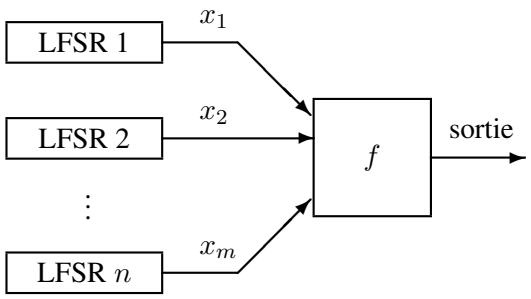
Remarque 1 *On a défini la notion de profil de complexité d'une suite s. C'est la suite $L_k = \mathcal{L}(s^k)$. Voilà donc un autre critère pour le caractère pseudo-aléatoire d'une suite: son profil de complexité devrait être assez proche de celui d'une suite véritablement aléatoire, c'est-à-dire $L_k \simeq k/2$.*

5.4 Cryptographie par flots moderne

On a vu que les LFSR ne sont pas cryptographiquement sûrs à cause de leur structure linéaire. Une méthode standard de chiffrement à flot consiste à les associer à une fonction non linéaire.

5.4.1 LFSR combinés, filtrés

(voir schéma ci-dessous)



On utilise m LFSR qui, à chaque top d'horloge, sortent chacun un bit x_i. Et les bits $x_1, \cdots, x_m$ constituent l'entrée d'une fonction booléenne qui sort un bit à chaque top d'horloge. Les polynômes de rétroaction des LFSR sont en général publics, ainsi que la fonction booléenne. Seules les initialisations des LFSR sont secrètes et constituent la clé (ce qui était impossible en utilisant un simple LFSR, puisqu'on a vu que si le polynôme de rétroaction est connu, alors en observant L bits consécutifs, on peut calculer tous les suivants).

Rappels sur les fonctions booléennes:

Définition 8 *Soit $V_m = {\mathbb{F}_2}^m$. On appelle* fonction booléenne *sur V_m toute application de V_m dans $\mathbb{F}_2$.*

On munit naturellement l'ensemble des fonctions booléennes des opérations induites par les opérations du corps $\mathbb{F}_2$.

Exercice 1 *Montrer que l'ensemble des fonctions booléennes sur V_m est un espace vectoriel de dimension 2^m sur $\mathbb{F}_2$, une base étant constituée des indicatrices (fonctions caractéristiques) des singletons $\{u\}$, avec $u \in \mathbb{F}_2^m$.*

Les fonctions coordonnées $x = (x_1, \ldots, x_m) \rightarrow x_i$ (que l'on notera simplement x_i) sont les fonctions booléennes les plus élémentaires, après les fonctions constantes 0 et 1.
Les produits de fonctions coordonnées sont aussi des fonctions booléennes sur V_m. On les appelle les *monômes*. Le degré du monôme $\prod_{i \in I} x_i$ est le cardinal $|I|$ de I. L'unique monôme de degré 0 est la fonction constante 1.

A chaque monôme correspond un mot $u \in V_m$ tel que ce monôme s'écrive:

$$\prod_{i=1}^{m} {x_i}^{u_i}.$$

Le degré de ce monôme est le poids de Hamming du mot u (son nombre de coordonnées non nulles).

Il existe plusieurs façons de noter le monôme $\prod_{i=1}^{m} {x_i}^{u_i}$. On peut le noter sous la forme x^u, où u désigne le mot $(u_1, \ldots, u_m)$.

Théorème 4 *L'ensemble des monômes constitue une base de l'espace vectoriel des fonctions booléennes sur V_m.*

Preuve:
Montrons d'abord que l'ensemble des monômes est une famille génératrice. En vertu de l'exercice 1, il suffit pour cela de montrer que l'indicatrice d'un singleton quelconque se décompose sur cette famille. Soit donc $a = (a_1, \ldots, a_m)$ un élément quelconque de V_m et 1_a l'indicatrice du singleton $\{a\}$; en développant l'égalité suivante:

$$1_a(x) = \prod_{i=1}^{m} (x_i + a_i + 1),$$

on obtient bien une décomposition de 1_a sur la famille des monômes.
Il reste à montrer que la famille des monômes est libre, ce qui est évident puisque son cardinal est égal à la dimension de l'espace vectoriel des fonctions booléennes.
$\diamond$

Définition 9 *L'écriture (unique) d'une fonction booléenne $f(x)$ comme combinaison linéaire à coefficients dans $\mathbb{F}_2$ de monômes s'appelle la* forme algébrique normale *de $f(x)$.*
Elle s'écrit:

$$f(x) = \sum_{u \in V_m} a_u \left(\prod_{i=1}^{m} {x_i}^{u_i} \right) = \sum_{u \in V_m} a_u \, x^u$$

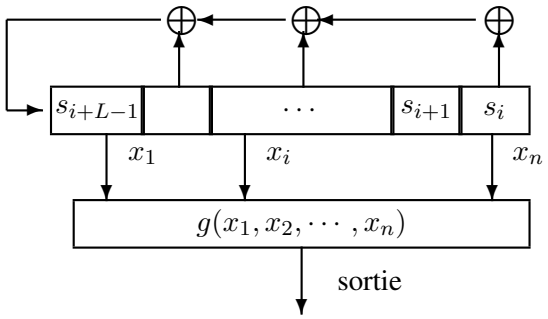
où, pour tout u, le coefficient a_u est dans $\mathbb{F}_2$.

Cette écriture, unique, correspond à la représentation de la fonction booléenne comme fonction polynomiale à m variables dont le degré relatif à chacune des variables est au plus 1 (d'où le nom de *normale*).
On appelle *degré* d'une fonction booléenne le degré global de sa forme algébrique normale.

> `Exemples:` L'indicatrice d'un singleton est toujours de degré m.
> L'indicatrice du singleton $\{(1, \ldots, 1)\}$ est sa propre transformée de Möbius.
> La transformée de Möbius de l'indicatrice du singleton $\{(0, \ldots, 0)\}$ est la fonction constante 1.

Une fonction booléenne utilisée comme fonction de combinaison doit être équilibrée (prendre le même nombre de fois les valeurs 0 et 1), car sinon la suite s engendrée elle-même ne prend pas le même nombre de fois les valeurs 0 et 1. Cela limite le degré de la fonction à $m - 1$. Enfin, puisque la suite générée par le système est périodique, elle est attaquable par l'algorithme de Berlekamp-Massey. Il faut donc qu'elle soit de complexité linéaire élevée. On démontre que la complexité linéaire d'une fonction $f(x_1, \cdots, x_m) = \sum_{u \in V_m} a_u \, (\prod_{i=1}^{m} {x_i}^{u_i}) = \sum_{u \in V_m} a_u \, x^u$ prenant en entrées des suites ML générées par des LFSR de longueurs $L_1, \cdots, L_m$ deux à deux distinctes et plus grandes que 2, est de complexité linéaire $f(L_1, \cdots, L_m)$, c'est à dire $\sum_{u \in V_m} a_u \, (\prod_{i=1}^{m} {L_i}^{u_i})$, *où le calcul de la somme est effectué dans* $\mathbf{R}$ *et non modulo 2* (voir la feuille d'exercices 2 pour une démonstration dans le cas où les L_i sont deux à deux premiers entre eux). On voit qu'il faut utiliser des fonctions de hauts degrés pour atteindre de hautes complexités linéaires.

LFSR filtrés On a un seul LFSR et à chaque top, l'état du registre (la suite des bits qu'il contient) ou un état partiel (une sous-suite) est l'entrée d'une fonction booléenne. On montre que ce système est équivalent au précédent: en effet, il revient à combiner n LFSR de même polynôme de connexion, dont les registres initiaux sont les n premiers registres de celui-ci; mais les attaques sur ce système



ne fonctionnent pas avec la même complexité que sur le système par fonction de combinaison qui lui est équivalent. Par exemple, la complexité linéaire est bornée par (et presque toujours égale à) $\sum_{i=1}^{m} \binom{L}{i}$, où m est le degré de f.

En effet, les équations suivantes calculent la sortie de ce générateur:

$$z_k = f(R_k) = f(A^k R_0)$$

Ces équations sont linéaires en les variables: $x_{i_1,i_2,\ldots,i_r} = x_{i_1} x_{i_2} \ldots x_{i_r}$ pour $r \leq m$, où $R_0 = [x_0, \ldots, x_{L-1}]$ est le registre initial, et le nombre de ces variables est bien $\sum_{i=1}^{m} \binom{L}{i}$.

Une autre approche consiste à résoudre les équations algébriques

$$z_k = f(R_k) = f(A^k R_0)$$

en les coordonnées de R_0 par des techniques du type "bases de Gröbner".

5.4.2 Générateurs avec contrôle d'horloge

L'idée est d'introduire de la non-linéarité dans une combinaison de LFSR en contrôlant l'horloge de l'un par l'autre.

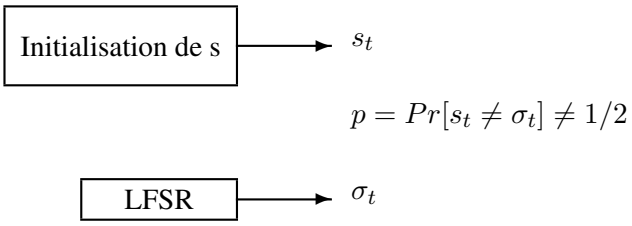
> `Exemple: le générateur rétrécissant (shrinking generator)`
>
> On prend deux LFSR R_1 et R_2. Les bits de sortie sont les bits de sortie de R_2, lorsque R_1 sort la valeur 1. Ce générateur fut proposé en 1993 par Coppersmith, Krawczyk, Mansour.

`Exemple: A5/1 le chiffrement à flot utilisé dans le GSM`

Il s'agit d'un algorithme propriétaire, qui n'a pas tardé à être connu de tous, et cryptanalysé. C'est une combinaison de quatre LFSR, dont l'un contrôle l'horloge, avec une fonction booléenne de degré deux.

5.4.3 Attaques par corrélation

Les attaques par corrélation sont les attaques les plus communes sur les générateurs pseudo-aléatoires. L'idée est de trouver un LFSR dont la sortie est corrélée avec la suite s considérée, à partir de la connaissance d'un certain nombre de bits de s.



Siegenthaler en 85 est le premier à décrire une attaque de ce type. Elle concerne le cas où la sortie du système est corrélée à l'un des LFSR le composant, ou à une combinaison de ceux-ci. Un exemple typique est le générateur de Geffe:

`Exemple: le générateur de Geffe`

On prend trois LFSR de longueurs maximales deux à deux premières entre elles L_1, L_2, L_3, et comme fonction booléenne de combinaison:

$$f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3.$$

Alors la période de la suite engendrée est égale à $(2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$ et sa complexité linéaire vaut $L = L_1L_2 + L_2L_3 + L_3$.

On voit facilement que, si $z(t)$ est la sortie de ce générateur à l'instant t, c'est-à-dire $z(t) = f(x_1(t), x_2(t), x_3(t))$, alors $z(t) = x_1(t)$ avec probabilité $3/4$ (et, de même, $z(t) = x_3(t)$ avec probabilité $3/4$). En effet, si $x_2 = 1$, alors $z = x_1$, et si $x_2 = 0$ alors $z = x_3$, mais $x_3 = x_1$ avec probabilité $1/2$!.

On peut alors monter l'attaque suivante: parcourir les valeurs possibles de la clé du LFSR_1 jusqu'à ce que sa sortie et la sortie de z coïncident dans $3/4$ des cas.

Dans un système combinant m LFSR on peut éviter ce type d'attaque en choisissant f non corrélée à l'ordre k pour k petit.

Définition 10 *On dit que f est non-corrélée à l'ordre k si, pour $X_1, \ldots, X_m$ des variables aléatoires binaires équidistribuées indépendantes, la variable aléatoire $f(X_1, \ldots, X_m)$ est indépendante des variables $\sum_{i \in I} X_i$, où I est un sous-ensemble de $\{1, \ldots, m\}$ de cardinal au plus égal à k.*

On dit que f est k-résiliente si f est non-corrélée à l'ordre k, et équilibrée.

Cela revient à dire que $d_H(f, \sum_{i \in I} x_i) = 2^{m-1}$, pour tout sous-ensemble I de cardinal au plus égal à k.

Pour étudier cette propriété, un outil important est la transformée de Walsh de f, i.e. la transformée de Fourier de $(-1)^f$. En effet, les valeurs de cette transformée calculent la distance de f aux fonctions affines. On montre que la propriété de k-résilience oblige à un compromis avec la recherche de fonctions de complexité linéaire élevé. En effet, si f est k-résiliente, alors le degré de f est au plus égal à $m - k$. D'autre part, la formule de Parceval (Plancherel?) montre qu'il existe toujours des fonctions affines corrélées à f; il faut donc choisir f de sorte que ces coefficients soient petits. On appelle cela la propriété de "non linéarité" de f. Soyons plus précis: le coefficient de corrélation de deux fonctions booléennes f et g est défini par:

$$c(f, g) = \frac{1}{2^m} \sum_{x \in \mathbb{F}_2^m} (-1)^{f(x)+g(x)} = 1 - \frac{d_H(f, g)}{2^{m-1}}.$$

On montre que:

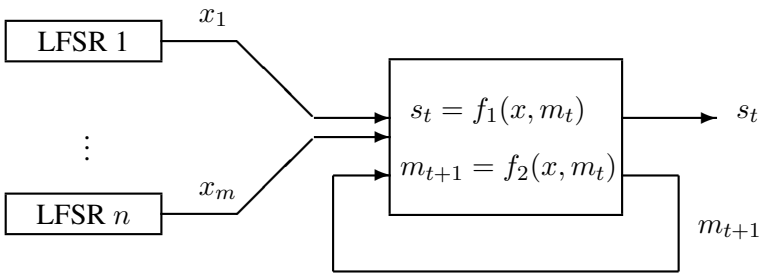
$$\sum_w c(f, l_w)^2 = 1, \text{ où } l_w(x) = w \cdot x.$$

La condition d'être non-corrélée à l'ordre k signifie: $c(f, l_w) = 0$ pour tout w de poids au plus égal à k. On voit bien que cela "force" les autres coefficients à être plus gros.

On peut interpréter une attaque par corrélation comme un problème de décodage. En effet, supposons connaitre les N premiers bits de sortie du générateur. On leur associe un élément $s \in \mathbb{F}_2^N$. Le LFSR concerné par la corrélation engendre, s'il est de longueur L, un sous-espace de $\mathbb{F}_2^N$ de dimension L, en faisant varier son initialisation, c'est-à-dire un code C de dimension L. Notre problème est donc de

trouver $c \in C$ qui rend minimal $d_H(s, c)$; c'est bien un problème de décodage. La méthode de Siegenthaler est en fait la recherche exhaustive. En 88, Meier et Staffelbach introduisent une "attaque par corrélation rapide" qui n'est autre qu'un algorithme de décodage itératif analogue à celui introduit en théorie des codes par Gallager dans les années 60. Ultérieurement d'autres techniques de décodage ont été proposées.

Pour éviter le problème des coefficients de corrélations, on introduit souvent de la mémoire dans un générateur combiné; cette idée, dûe à Rueppel, rend plus difficile l'analyse des corrélations, sans les exclure. Typiquement, un bit de mémoire est produit par l'addition de deux suits binaires, exécutée dans $\mathbb{N}$ (c'est la retenue!). Exemple actuel: le système cryptographique de Bluetooth.



5.4.4 Autres générateurs symétriques

Parmi les générateurs non basés sur des LFSR, on peut citer:

- Les systèmes de chiffrement par blocs en mode OFB, CFB
- SEAL (Coppersmith and Rogaway 1993)
- RC4: algorithme propriétaire de la société RSA-Security. Cet algorithme est largement employé (par exemple dans la technologie de WIFIE 802.4). Il génère un permutation pseudo-aléatoire.

Chapter 6

Les schémas de chiffrement par blocs

Dans ce chapitre, nous allons décrire la structure de quelques systèmes itératifs, en particulier les deux standards DES et AES, ainsi que les deux principales attaques connues, qui sont la cryptanalyse différentielle et la cryptanalyse linéaire.

6.1 Description de quelques systèmes

6.1.1 Les schémas de Feistel et le DES

Dans un schéma de Feistel, le clair est de longueur paire et découpé en une moitié gauche L et une moitié droite R. Le $i$ème tour du schéma prend en entrée un bloc (L_{i-1}, R_{i-1}) et le transforme (en faisant intervenir la $i$ème sous-clé K_i) en un bloc (L_i, R_i). Le premier tour prend en entrée le bloc (L, R) et le dernier tour produit le chiffré (L', R'). La relation entre (L_{i-1}, R_{i-1}) et (L_i, R_i) est:

$$L_i = R_{i-1},\ R_i = L_{i-1} + f(R_{i-1}, K_i)$$

où f est un chiffrement produit.

Un chiffrement de Feistel est inversible, que f soit une bijection ou non. En effet, on a

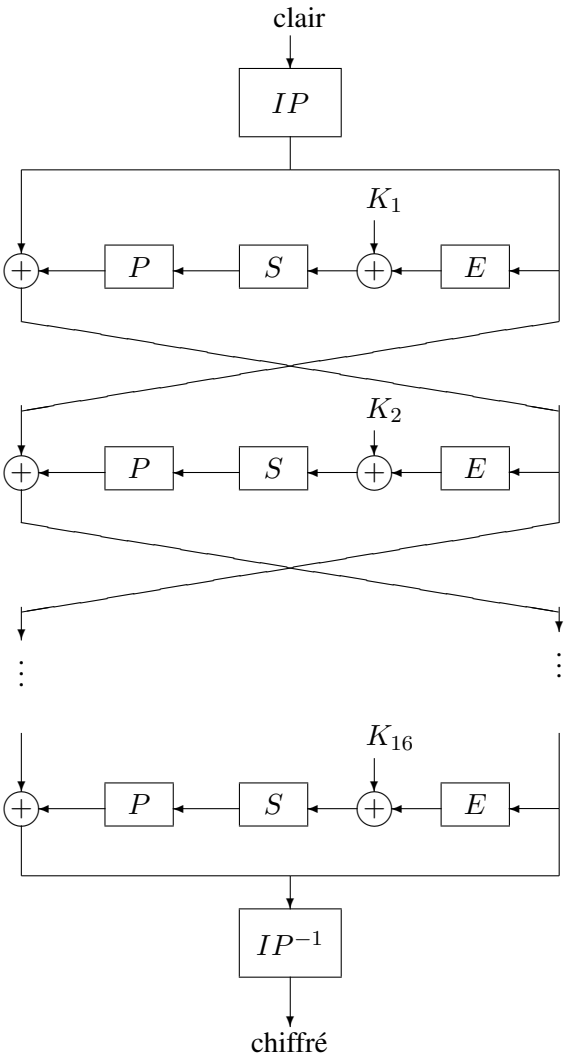
$$R_{i-1} = L_i,\ L_{i-1} = R_i + f(R_{i-1}, K_i).$$

Ainsi, le permuté (R, L) du clair s'obtient à partir de celui (R', L') du chiffré en lui appliquant un chiffrement de Feistel de même schéma et en prenant comme liste de sous-clés de déchiffrement la même que celle des sous-clés de chiffrement, mais dans l'ordre inverse.

Le DES qui a été le standard (proposé par le National Bureau of Standards américain et développé par IBM) utilisé de 1977 à 2000 dans le monde entier, est un schéma de Feistel. L'algorithme de chiffrement est complètement spécifié mais fait intervenir des *boites S* dont les critères de conception n'ont pas été rendus publics. Certains pensent que la NSA (National Security Agency) qui avait participé à l'opération avait introduit des *trappes* lui permettant de décrypter plus facilement les messages, mais cela n'a jamais pu être prouvé puisqu'aucune cryptanalyse sensiblement plus efficace que l'attaque exhaustive n'a pu être trouvée[1]. En janvier 98, une attaque exhaustive a été réalisée en 39 jours sur 10 000 pentiums en parallèle. Et en juillet 98, un autre attaque a pris 56 heures avec une machine dédiée de 250 000 dollars. L'adoption de l'AES s'imposait donc.

Le DES opère par blocs de 64 bits avec une clé de 56 bits (complétés de 8 bits de redondance). Le clair est permuté par une permutation initiale IP puis 16 itérations sont effectuées. Les sous-clés sont de 48 bits. Après les 16 tours, on applique IP^{-1}. La fonction de chiffrement f qui opère à chaque tour est le produit (voir schéma à la page suivante) d'une fonction (on dit aussi boite) d'expansion E qui est une application linéaire de F_2^{32} dans F_2^{48} ne faisant pas intervenir la clé, de l'ajout bit à bit de la clé, d'une fonction de substitution (boite S) qui est une application non linéaire de F_2^{48} dans F_2^{32} ne faisant pas intervenir la clé et d'une permutation P des bits (qui est donc une fonction linéaire de F_2^{32} dans F_2^{32}). La boite S, qui est la plus complexe à spécifier et qui assure la fonction de confusion, essentielle à la sécurité du DES, est en fait un produit de 8 fonctions de substitution de F_2^6 dans F_2^4 données explicitement. Les applications coordonnées de ces sous-fonctions de substitution ont été choisies (1) pour permettre au système de résister à l'attaque différentielle (voir plus loin) qui était connue des concepteurs du DES mais tenue secrète et (2) de façon qu'elles soient à grandes distances de Hamming de l'ensemble des fonctions affines (cela a contribué au fait que le système résiste bien à l'attaque linéaire qui n'était pourtant pas connue des concepteurs). Rappelons que la distance de Hamming entre deux fonctions booléennes est égale au nombre des vecteurs en lesquels elles prennent des valeurs différentes. Les applications E et P assurent la diffusion.

[1] La même suspicion n'existe pas concernant l'AES qui a résulté d'un appel d'offre.



Le *mode opératoire* du DES peut être le mode ECB (Electronic Code Book): chaque bloc de 64 bits du clair est chiffré avec la clé (le i-ème bloc de chiffré se déduit donc du $i$ème bloc de clair par la relation $c_i = E_K(m_i)$) préférable quand il y a un risque d'erreur de transmission du chiffré (malgré l'utilisation éventuelle de codes correcteurs d'erreurs) ou CBC (Cipher Block Chaining) $c_i = E_K(m_i \oplus c_{i-1})$ et $c_1 = E_K(m_1 \oplus IV)$ où IV est un *vecteur d'initialisation* choisi aléatoirement, préférable dans tous les autres cas (mais à éviter s'il y a risque d'erreur de transmission car une erreur sur un bloc se propagerait à tous les blocs suivants): chaque

bloc de clair influence tous les chiffrés suivants; cela interdit l'attaque dite par dictionnaire qui permet au possesseur d'un stock de couples (clair, chiffré) de savoir déchiffrer tous les blocs de chiffrés de son stock (cette attaque élémentaire est efficace car il n'est pas rare que des fragments de messages particuliers tels que les en-têtes se répètent). De plus, cette méthode de chiffrement a l'avantage d'être probabiliste (deux chiffrés différents d'un même message sont différents).

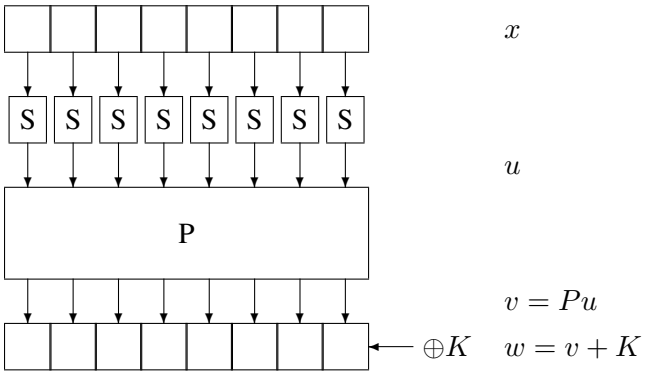
6.1.2 Le triple DES

Pour palier à la faiblesse du DES concernant sa longueur de clé (56 bits), on utilise souvent une version renforcée appelée *triple DES* ou 3DES. Ce n'est pas la panacée car cette version est trop lente, comparée à un AES par exemple.

Si E_K, D_K sont les fonctions de chiffrement et de déchiffrement du DES, on chiffre par: $x \rightarrow E_{K_1}(D_{K_2}(E_{K_1}(x)))$, ce qui double la longueur de la clé. La solution plus simple: $x \rightarrow E_{K_1}(E_{K_2}(x))$ n'est pas satisfaisante car elle est vulnérable à une attaque dite *de l'homme du milieu*: si $y = E_{K_1}(E_{K_2}(x))$, et si on connait le couple (x, y), on peut calculer et stocker les $E_K(x), K$ puis calculer les $D_T(y)$ jusqu'à trouver une collision (cf le paradoxe des anniversaires).

6.1.3 Les schémas de substitution/permutation (S/P)

On appelle ainsi les schémas itératifs dont les tours sont construits de la façon suivante:



Si x est l'entrée du tour, x est découpé en sous-blocs d'égale longueur $x_1, x_2, \ldots$ auxquels on applique une substitution S (éventuellement il y a autant de substitutions que de sous-blocs). Si $u_i = Sx_i$, on concatène les u_i pour former u, à qui on applique une permutation P (éventuellement une transformation linéaire plus

complexe). On note $v = Pu$. La dernière étape du tour est l'ajout de la clé de tour K à v. Notons $w = v + K$.

Le premier tour - supposons qu'il y en r - est précédé de l'ajout d'une clé supplémentaire K_0 pour éviter que l'attaquant déchiffre le premier tour jusqu'à l'ajout de K_1. L'algorithme RIJNDAEL, choisi pour être le nouveau standard du chiffrement par blocs AES, est de ce type.

6.1.4 L'AES

L'Advanced Encryption Standard a fait l'objet d'un appel d'offre datant de 1997. Il s'agissait de remplacer le DES dont la taille des clés était devenue trop petite pour les performances des ordinateurs modernes. Les spécifications étaient une longueur de blocs de 128 bits (ou de 256 bits) et une longueur de clé paramétrable: 128 ou 192 ou 256 bits. Parmi les 15 candidats, le candidat retenu (en 2000) se nomme RIJNDAEL (mais on l'appelle simplement l'AES). Il est dû à deux chercheurs Belges, Rijmen et Daemen. C'est un chiffrement itératif, mais contrairement à 9 autres candidats, ce n'est pas un chiffrement de Feistel. C'est un chiffrement par substitution-permutation: à chaque tour, le chiffré produit par le tour précédent subit une substitution non-linéaire qui assure la confusion puis une permutation linéaire qui assure la diffusion, puis la clé du tour est ajoutée bit à bit. Le nombre de tours est de 10 pour une clé de 128 bits et de 14 pour une clé de 256 bits.

La fonction de diffusion agit sur les 16 octets de l'entrée en les permutant puis en appliquant la même application linéaire sur chaque bloc de 4 octets consécutifs. Chaque octet est identifié à un élément du corps $F_{2^8} = F_{256}$ à 256 éléments. Le polynôme irréductible utilisé pour la construction de ce corps est $X^8 + X^4 + X^3 + X + 1$. Ce n'est pas un polynôme primitif mais les concepteurs l'ont choisi pour accélérer les calculs. L'identification entre les éléments de ce corps et les octets se fait classiquement: on choisit une base du corps vu comme espace vectoriel de dimension 8 sur F_2; chaque élément du corps peut ainsi être identifié à un vecteur binaire de longueur 8, c'est à dire à un octet. L'application linéaire est définie par sa matrice 4×4 à coefficients dans le corps F_{256}:

$$\begin{bmatrix} \alpha & \alpha+1 & 1 & 1 \\ 1 & \alpha & \alpha+1 & 1 \\ 1 & 1 & \alpha & \alpha+1 \\ \alpha+1 & 1 & 1 & \alpha \end{bmatrix}$$

où α est un élément primitif. Elle agit donc principalement au niveau global des octets (mais elle agit quand même au niveau des bits via les multiplications par les

éléments de ce corps).

La fonction de substitution (la boite S): chaque octet est considéré comme un élément du corps F_{256}. La boite S est constituée de 16 boites identiques consécutives agissant chacune sur un octet. Chaque boite S_i consiste en l'application $F : x \in F_{256} \mapsto x^{254} \in F_{256}$. Notons que si $x = 0$ alors $x^{254} = 0$ et si $x \neq 0$ alors $x^{254} = \frac{1}{x}$. Le résultat de cette transformation subit ensuite une application affine. Cette boite S permet au système de résister à l'attaque différentielle et à l'attaque linéaire. La raison de ce choix est que la fonction $F : x \in F_{256} \mapsto x^{254} \in F_{256}$ est telle que l'équation $F(x) + F(x + a) = b$ admet au plus 4 solutions (cela évite l'existence de différentielles de probabilités élevées) et que l'équation $a \cdot X \oplus b \cdot F(X) = 0$ admet un nombre de solutions proche de 2^{128} pour tout $a \neq 0$ et tout $b \neq 0$ (cela rend couteuse l'attaque linéaire).

6.1.5 Cryptanalyse

Les deux principales méthodes connues de cryptanalyse des chiffrements par blocs symétriques sont la cryptanalyse différentielle et la cryptanalyse linéaire. Elles exploitent toutes deux des comportements statistiques non uniformes dans le processus de chiffrement. La cryptanalyse différentielle date de 1990 et est dûe à Biham et Shamir. La cryptanalyse linéaire date de 1992 et est dûe à Matsui.

Une attaque générique sur les chiffrement itératifs: appelons x le texte clair, y son chiffré. En posant $x_0 = x$ et $x_i = F(x_{i-1}, K_i)$ pour tout $i = 1, \ldots, r$, on a $y = x_r$. Le but de l'attaque est de déterminer la valeur de K_r, puis, en utilisant le chiffrement de x en $x_{r-1} = F^{-1}(y, K_r)$ qui est un chiffrement à $r - 1$ tours, de calculer K_{r-1}, et ainsi de suite.

Nous nous concentrons donc sur le calcul de K_r. On suppose qu'il existe une corrélation entre x et x_{r-1}. L'attaque consiste alors à itérer pour un grand nombre de couples clairs-chiffrés (x, y) les étapes suivantes:

Pour toutes les valeurs possibles k de K_r:

- Calculer $z := F^{-1}(y, k)$.
- Calculer $p := \text{cor}(x, z)$.
- Si $p = \text{cor}(x, x_{r-1})$, incrémenter un compteur associé à la valeur k.

Lorsque la boucle passe par la valeur $k = K_r$, bien sûr, la condition $p = \text{cor}(x, x_{r-1})$ est réalisée et le compteur de K_r est incrémenté avec une fréquence non uniforme. On espère que, lorsque une valeur $k \neq K_r$ est choisie, la variable z se comporte

comme une variable indépendante de x; alors la condition $p = \text{cor}(x, x_{r-1})$ est réalisée avec une probabilité uniforme. Finalement, le compteur incrémenté non uniformément est celui de K_r.

Dans la pratique, on ne parcourt pas toutes les valeurs possibles pour K_r ce qui serait impossible mais on isole la partie de K_r qui intervient vraiment dans le calcul de $\text{cor}(x, z)$. Cela permet de calculer K_r petits morceaux par petits morceaux.

La cryptanalyse différentielle s'intéresse à l'évolution les différences $x_i + x_i^*$ pour deux clairs x, x^*. Il s'agit d'une attaque à clairs choisis. Notons

$$P_{\alpha,\beta} := \text{prob}(x_{r-1} + x_{r-1}^* = \beta \mid x + x^* = \alpha)$$

et supposons avoir déterminé une valeur de (α, β) telle que cette probabilité soit particulièrement élevée.

On itère, pour un grand nombre de couples clairs-chiffrés (x, y) et (x^*, y^*) tels que $x + x^* = \alpha$, les étapes suivantes:

Pour toutes les valeurs possibles k de K_r:

- Calculer $z := F^{-1}(y, k)$ et $z^* := F^{-1}(y^*, k)$.
- Si $z + z^* = \beta$, incrémenter un compteur associé à k.

Le compteur le plus souvent incrémenté est celui de K_r. Notons que l'attaque est d'autant plus efficace que la probabilité $P_{\alpha,\beta}$ s'éloigne de la probabilité uniforme, soit $1/2^n$. On peut restreindre l'ensemble des valeurs de k parcouru à celles qui influent sur la valeur de $z + z^*$ (en effet, si deux valeurs de k donnent toujours le même $z + z^*$, elles sont également incrémentées).

Il faut remarquer que, dans les étapes linéaires du chiffrement, le comportement des différences est complètement déterminé puisque, si f est linéaire, $f(x_i + x_i^*) = f(x_i) + f(x_i^*)$. De même, l'ajout d'une clé ne change pas la différence: $(x_i + K) + (x_i^* + K) = x_i + x_i^*$. Par contre, si f n'est pas linéaire, on a seulement $x_i + x_i^* = 0 \Rightarrow f(x_i) + f(x_i^*) = 0$, alors que, si $x_i + x_i^* = \alpha \neq 0$, la valeur de $f(x_i) + f(x_i^*)$ n'est pas déterminée.

Il reste à éclaircir deux points: le calcul de $P_{\alpha,\beta}$, et la façon de restreindre la boucle sur k.

Exemple: les systèmes S/P. On peut calculer les probabilités $P_{\alpha,\beta}$ *relatives à* S. Supposons que $S : \mathbb{F}_2^s \to \mathbb{F}_2^s$. On calcule les entrées de la matrice D de taille $2^s \times 2^s$

$$D[\alpha, \beta] = \text{card}\{(x, x^*) \in (\mathbb{F}_{2^s})^2 \mid x + x^* = \alpha \text{ et } Sx + Sx^* = \beta\}.$$

Alors, pour les entrées et sorties de S, on a:

$$p_{\alpha,\beta} := \text{prob}(Sx + Sx^* = \beta \mid x + x^* = \alpha) = D[\alpha, \beta]/2^s.$$

On peut donc suivre l'évolution des différences dans l'algorithme, en découpant par sous-blocs:

$$\text{prob}(v + v^* = \beta \mid u + u^* = \alpha) = \prod_i \text{prob}(v_i + v_i^* = \beta_i \mid u_i + u_i^* = \alpha_i),$$

et en tenant compte de l'action de la permutation P sur les β_i. On dit qu'une boite S est *active* au l-ème tour si la différence en entrée (et donc en sortie) est non nulle. Il suffit alors de faire varier les bits de la clé K_r dont l'image par P^{-1} correspond à une boite active au r-ième tour. Prenons en exemple l'algorithme jouet B_{32}, pour lequel $s = 4$ et P est le décalage de deux positions vers la droite. Le meilleur coefficient de D est 10, obtenu pour $\alpha_1 = 1111$ et $\beta_1 = 1001$. Essayons $\alpha = (\alpha_1, \underline{0}, \underline{0}, \ldots, \underline{0})$, où $\underline{0} = 0000$. α devient $\beta = (\beta_1, \underline{0}, \underline{0}, \ldots, \underline{0})$, avec probabilité $10/16$, puis par P, $(0010, 0100, \underline{0}, \ldots, \underline{0})$. Deux boites sont actives au deuxième tour. Pour retrouver K_2, il faut donc parcourir les 2^8 valeurs possibles pour $(k[3], k[4], \ldots, k[10])$ en prenant les autres bits de k nuls. Une autre possibilité, qui permet d'attaquer un nombre arbitraire de tours, est de considérer $\alpha_1 = 0001$ qui se transforme en $\beta_1 = 0100$ avec probabilité $6/16$. En effet, $P\beta_1 = \alpha_1$ donc une seule boite est active quel que soit le nombre de tours et $P_{\alpha,\beta} = (6/16)^{r-1}$.

La cryptanalyse linéaire s'intéresse aux relations linéaires entre les bits au cours de l'algorithme. C'est une attaque à clair connu. Pour $a \in \mathbb{F}_2^n$ et $x \in \mathbb{F}_2^n$, notons $a \cdot x = a_1x_1 + \cdots + a_nx_n$. Nous présentons deux versions de l'attaque: l'attaque sur le chiffrement global et l'attaque sur le dernier tour. Notons

$$P_{a,b,c} := \text{prob}(a \cdot x + b \cdot y + c \cdot K = 0)$$

et supposons avoir déterminé une valeur de (a, b, c) tels que cette probabilité soit particulièrement élevée.

Dans l'attaque sur le chiffrement global, on calcule, pour un grand nombre de couples clairs-chiffrés (x, y), la valeur de $a \cdot x + b \cdot y$. Si cette valeur vaut 0 plus souvent que 1, on en conclut que $c \cdot K = 0$. Sinon, c'est que $c \cdot K = 1$. On obtient ainsi une relation linéaire entre les bits de K; si la longueur de K est m, il faut en trouver m linéairement indépendantes pour déterminer K, ou du moins suffisamment pour autoriser l'attaque exhaustive (dans un S/P, si les clés de tour sont indépendantes, $m = (r + 1)n$. Souvent une algorithme de cadencement des clés dérive les clés de tour d'une même clé K).

Une méthode plus efficace consiste à attaquer le dernier tour comme dans le cas de l'attaque différentielle: notons maintenant

$$P_{a,b} := \mathrm{prob}(a \cdot x + b \cdot x_{r-1} = 0)$$

et supposons que $P_{a,b}$ soit assez éloigné de $1/2$.

On itère, pour un grand nombre de couples clairs-chiffrés (x, y) les étapes suivantes:

Pour toutes les valeurs possibles k de K_r:

- Calculer $z := F^{-1}(y, k)$
- Si $a \cdot x + b \cdot z = 0$, incrémenter un compteur associé à k.

Pour un assez grand nombre de couples, un compteur est particulièrement élevé ou bas, c'est celui de K_r. Notons que, si pour deux valeurs k et k', $b \cdot z = b \cdot z'$, les compteurs de k et k' seront également incrémentés. On peut donc restreindre l'ensemble des valeurs de k à parcourir, quitte à ne déterminer qu'une partie de K_r (et cela est souhaitable car on ne peut pas en général parcourir toutes les valeurs possibles pour K_r). On doit alors changer de (a, b).

Afin de calculer $P_{a,b}$, on se ramène à l'analyse de ce qui se passe dans l'étape non linéaire de chaque tour, via le *Piling-up lemma*:

Lemme 5 *Soit $Z_1, \ldots, Z_r$ des variables aléatoires indépendantes booléennes, avec $\mathrm{prob}(Z_i = 0) = p_i$. Alors,*

$$\mathrm{prob}(Z_1 + \cdots + Z_r = 0) = 1/2 + 2^{r-1} \prod_{i=1}^{r} (p_i - 1/2).$$

Preuve: Se démontre par récurrence à partir de $r = 2$. L'évènement $Z_1 = Z_2$ est la réunion disjointe de $Z_1 = Z_2 = 0$ et de $Z_1 = Z_2 = 1$. Donc $\mathrm{prob}(Z_1 + Z_2 = 0) = p_1 p_2 + (1 - p_1)(1 - p_2) = 1/2 + 2(p_1 - 1/2)(p_2 - 1/2)$.

□

Ce lemme nous permet de calculer $P_{a_0, a_{r-1}}$, si on connait les probabilités $\mathrm{prob}(a_{i-1} \cdot x_{i-1} + a_i \cdot x_i = 0)$, pour $i = 1, \ldots, r-1$. En effet, $a_0 \cdot x_0 + a_{r-1} \cdot x_{r-1} = (a_0 \cdot x_0 + a_1 \cdot x_1) + \cdots + (a_{r-2} \cdot x_{r-2} + a_{r-1} \cdot x_{r-1})$, et $x_0 = x + K_0$.

Exemple: les systèmes S/P. On peut calculer les probabilités $P_{a,b}$ *relatives à* S. On calcule pour cela la matrice L de taille $2^s \times 2^s$

$$L[a, b] := \mathrm{card}\{x \in \mathbb{F}_2^s \mid a \cdot x + b \cdot Sx = 0\}.$$

Alors, pour les entrées et sorties de S, on a:

$$p_{a,b} := \text{prob}(a \cdot x + b \cdot Sx = 0) = L[a,b]/2^s.$$

On peut donc suivre l'évolution des combinaisons linéaires dans l'algorithme, en découpant par sous-blocs, en tenant compte de l'action de la permutation P, et grâce au lemme précédent, et calculer $P_{a,b}$.

Les matrices D et L associées à Rijndael s'interprètent en termes d'équations à coefficients dans $\mathbb{F}_{256}$. Notons $i(z)$ l'application définie par $i(0) = 0$ et $i(z) = 1/z$ pour $z \in \mathbb{F}_{256}^*$. La boite S de Rijndael vaut $Sz = f(i(z)) + c$, où f est une application linéaire inversible. Donc $Sz + Sz^* = f(i(z)) + f(i(z^*)) = f(i(z) + i(z^*))$, et

$$D[\alpha,\beta] = \text{card}\{(x,x^*) \mid x + x^* = \alpha \text{ et } i(z) + i(z^*) = f^{-1}(\beta)\}.$$

Notons $\beta' = f^{-1}(\beta)$. Si $z = 0$, $z^* = \alpha$ et la condition est réalisée ssi $\alpha^{-1} = \beta'$. Si $z \neq 0$ et $z^* \neq 0$, on doit résoudre l'équation $1/z + 1/(z+\alpha) = \beta'$ qui est une équation de degré deux en z. Dans le corps $\mathbb{F}_{256}$, elle a donc au plus deux solutions. Ainsi, $D[\alpha,\beta] \leq 4$.