Testeurs, problemes de reconstruction univaries et multivaries, et application a la cryptanalyse du DES.

Cédric Tavernier

année : 2004

N° attribué par la bibliothèque

THÈSE

présentée à

L'ÉCOLE POLYTECHNIQUE

pour obtenir le titre de

DOCTEUR EN SCIENCES

Spécialité

Codes correcteur d'erreurs, cryptographie

soutenue par

Cédric TAVERNIER

le 15 Janvier 2004

Titre

Testeurs, problèmes de reconstruction univariés et multivariés, et application à la cryptanalyse du DES

Jury

Président	Grigory Kabatiansky
Directrice	Pascale Charpin
Co-directeur	Daniel Augot
Rapporteur	Gilles Zemor
Examinateurs	Henri Gilbert
	Andreas Enge

Remerciements

Tout d'abord je tiens à remercier Pascale Charpin et Daniel Augot de m'avoir donné la chance d'effectuer cette thèse au projet codes, et de m'avoir aussi bien dirigé dans mes recherches. Je remercie tous les autres membres de mon Jury de thèse. Je remercie également tous les autres permanents du projet codes: Nicolas Sendrier, Anne Canteaut, Jean-Pierre Tillich, Christelle Guiziou-Cloitre, qui m'ont permis d'effectuer une thèse dans des conditions si favorables. Je ne pourrai pas oublier les conditions privilégiées qui m'ont été offerte pour préparer cette thèse. Je remercie Anne Canteaut pour m'avoir appris la programmation en langage c, sans qui, je n'aurais pas pu faire la moindre expérience. Je remercie Grisha Kabatiansky pour toute l'aide mathématique qu'il m'a apporté. Je remercie Eric Filiol qui m'a très rapidement motivé pour les applications cryptographiques. Je n'oublie pas tous les thésards qui m'ont bien souvent aidé à dépasser mes lacunes informatiques. Un grand merci à Matthieu Finiaz pour sa patience et pour toute l'aide qu'il m'a apporté, tant sur un plan des connaissances que sur un plan matériel. Je remercie enfin d'une manière générale tous les chercheurs que j'ai cotoyé et qui m'ont enrichi de leur précieuses remarques. Je remercie également ma femme pour sa patience et pour tous les week-end sacrifiés.

Table des matières

Introduction

Les moyens de transmettre de l'information se sont considérablement accrus, ces dernières années, l'informatique, Internet, la téléphonie sont présents dans tous les secteurs de l'activité économique. En conséquence, la cryptographie est présente dans la vie quotidienne, comme par exemple dans les cartes de crédit, le paiement sur Internet, elle peut être aussi un acteur économique important lorsque dans un climat de compétition économique, on cherche à chiffrer des informations qui doivent rester secrètes. Il est donc nécessaire de développer des systèmes solides capables de résister à d'éventuels attaquants. Par ailleurs, la cryptographie n'est pas une discipline isolée, nous verrons que les objets que l'on étudiera font partie de la théorie des codes correcteurs d'erreurs.

Ainsi nous nous intéresserons dans ce document à deux classes très importantes de cryptosystèmes: les systèmes de chiffrement par bloc qui sont à clef secrète, et les systèmes à clef publique tels que les systèmes construits à partir de courbes elliptiques. Pour la première classe de cryptosystème, notre but sera de faire une analyse cryptographique de leur sécurité. Nous développerons dans cette optique différents algorithmes exploitant des faiblesses éventuelles de certains systèmes de chiffrement par bloc. On s'intéressera à une classe de problèmes liée au problème de reconstruction, une classe de problèmes liée à la correction d'erreurs, c'est à dire toute la théorie qui consiste à transmettre correctement de l'information à travers un canal bruité. L'un des objectifs est d'adapter des algorithmes de décodage aux paramètres cryptographiques, on sera aussi amené à considérer différents canaux de transmission qui correspondent aux problèmes cryptographiques rencontrés. Ces algorithmes seront alors systématiquement appliqués à un système bien connu, le DES. Les applications et les résultats de simulation tiendront alors une grande place dans ce document.

Une autre grande classe de cryptosystèmes concerne les systèmes à clef publique. Depuis quelques années, des systèmes tels que les courbes elliptiques, les courbes hyperelliptiques de genre 2 ou 3 ont montré qu'ils étaient très performants sur beaucoup de points de vue. Ces courbes sont aussi très importantes pour construire des codes géométriques qui possèdent des bonnes propriétés. Il s'avère que beaucoup de ces courbes proviennent d'une classe plus importante de courbes qui sont appelées courbes modulaires. Nous nous intéresserons dans cette

partie à la construction de courbes modulaires en vu de produire des cryptosystèmes performants.

Les travaux présentés dans cette thèse s'articulent autour de trois thèmes: le développement d'algorithmes de décodage, destinés à exploiter les faiblesses éventuelles des cryptosystèmes, différentes cryptanalyses et en particulier une analyse cryptographique du DES, et la construction de courbes qui forment de très bon cryptosystèmes. Le lecteur trouvera des rappels de théorie des codes correcteur d'erreurs, ainsi que des rappels de probabilité nécessaires à la bonne compréhension de ce document.

Ainsi dans le premier chapitre nous ferons des rappels de théorie des codes correcteurs d'erreurs, on décrira la problématique liée à la transmission de l'information à travers un canal bruité et à la correction d'erreurs, nous introduirons alors les codes qui sont centraux dans ce document, c'est-à-dire les codes de Reed-Solomon et les codes de Reed-Muller. On fera une introduction au problème de reconstruction afin de mettre en évidence la similitude avec le problème de décodage dans le cadre des codes de Reed-Solomon et des codes de Reed-Muller. On replacera enfin ces problèmes dans le contexte cryptographique qui nous intéresse, c'est-à-dire le décodage local [15] en grandes longueurs.

Pour bien comprendre à quels types d'objets nous aurons à faire face, on a consacré le second chapitre à la description des systèmes de chiffrement par bloc, et en particulier à la description du DES et du système de Knudsen et Nyberg [14]. On décrira alors dans le chapitre 3 deux cryptanalyses bien connues dont le principe repose sur des problèmes de reconstruction, soit encore des problèmes de décodage par liste en grandes longueurs. Les longueurs des mots de codes considérés sont si grandes qu'il est impossible d'avoir accès aux mots tout entier, on considérera donc toujours une fraction des éléments qui composent ces mots. C'est à dire que l'on explicitera une propriété statistique sur ce mot de code, d'où la nécessité d'introduire quelques notions de probabilités pour pouvoir aborder le problème de l'échantillonnage.

Le chapitre 4 est une brève introduction aux probabilités. Le but est surtout de donner les outils nécessaires à la bonne compréhension des règles d'échantillonage. On aura par exemple souvent besoin d'évaluer la taille de l'échantillon nécessaire pour mesurer d'une manière fiable une certaine propriété statistique.

Le chapitre 5 est consacré au problème du test. Cette problématique se situe dans le contexte des polynômes univariés. On a repris dans ce chapitre les travaux de M. Sudan [44]. Les théoriciens de la complexité se sont posé la question suivante: si $f : \mathbb{F} \to \mathbb{F}$ désigne une fonction sur un corps $\mathbb{F}$, alors n'est-il pas plus facile de tester s'il existe au moins un polynôme de degré au plus k qui coïncide avec f sur une fraction d'au moins δ de ses entrées sans pour autant reconstruire

ce ou ces polynômes, plutôt que de construire directement d'éventuels polynômes satisfaisant cette même condition? Les tests qui figurent dans la thèse de Sudan ne sont applicables que lorsque le corps considéré est un corps premier du type $\mathbb{F}_p$. Nous proposerons une extension de ces résultats aux corps finis du type $\mathbb{F}_q$, $q = 2^t$. Nous étudierons ces tests dans divers contextes et nous mettrons en avant les similitudes de ces problèmes avec les problèmes de reconstruction. Nous verrons que ces tests restent bien faibles du point de vue de la complexité face à l'algorithme de Guruswami-Sudan [17]. En particulier ces tests sont inapplicables pour des paramètres cryptographiques. Aussi nous avons tenté une expérience en appliquant directement l'algorithme de Sudan sur le DES.

Dans le chapitre 6 nous passerons du contexte des polynômes univariés aux polynômes multivariés. Sudan a décrit dans sa thèse [44] des tests similaires aux tests dans le contexte univarié. Ces tests multivariés sont applicables sur les corps premiers du type $\mathbb{F}_p$. Nous généraliserons ces tests aux corps finis du type $\mathbb{F}_q$, $q = 2^t$. La constatation est la même que pour les tests univariés, ces tests ne permettent pas de faire une analyse cryptographique des systèmes de chiffrement par bloc. On est donc passé du test au décodage. Nous avons repris les travaux de O. Goldreich, R. Rubinfeld, M. Sudan [16] concernant le problème de reconstruction multivariée en grandes longueurs. En particulier nous nous sommes intéressés au problème du décodage pour le Reed-Muller d'ordre 1 en grandes longueurs. O.Goldreich, R.Rubinfeld, M.Sudan donnent un algorithme de décodage pour le Reed-Muller d'ordre 1 qui est applicable non seulement au canal binaire symétrique, mais aussi au canal adversaire, la complexité de cet algorithme est imprécise dans l'article [16], on sait simplement que la complexité est polylogarithmique en la longueur du code et polynomiale en le poids de l'erreur normalisé. Nous ferons donc une analyse plus précise de cet algorithme dans différents types de canaux et nous proposerons une variante de cet algorithme dont la complexité est proche des bornes prévues par le théorie de l'information. Nous étendrons ensuite cet algorithme au Reed-Muller d'ordre 2. Nous verrons dans quel contexte cet algorithme donne les meilleurs résultats. On a jusqu'ici étudié des algorithmes de décodage adaptatifs, ce sont des algorithmes qui au cours de leur processus "posent" des questions à la fonction f que l'on considère puis qui agissent en fonction de la réponse. On montrera que par de simple transformations linéaires bijectives, il n'est pas possible de transformer notre algorithme en algorithme non adaptatif. Nous présenterons des résultats de simulations, puis nous appliquerons ces méthodes de décodage pour approximer certaines sorties du DES.

Enfin on fera dans le chapitre 7 une digression sur les courbes modulaires. Cette partie est davantage un survey des idées développées dans [18]. Notre but ici est de construire des courbes modulaires et de calculer le cardinal de leur jacobienne. On sait que les courbes modulaires sont utiles pour construire de bons codes géométriques [8], [26]. Depuis peu, on sait que toutes les courbes el-

liptiques sont modulaires, et il est conjecturé que toutes les variété abéliennes à multiplication réelle proviennent des courbes modulaires. Donc les courbes modulaires représentent une grande famille de courbes. De plus on sait que les courbes elliptiques et les courbes hyperelliptiques de genre 2, ou 3 forment de très bon cryptosystèmes à clef publique. On verra de quelle manière on peut développer un algorithme de construction de courbes qui permet de se restreindre aux courbes modulaires de genre 2 ou 3.

Chapitre 1

Rappels sur la théorie algébrique des codes

Ce chapitre contient un certain nombre de rappels bien connus, un ensemble de définitions et propriétés utiles pour la suite. Cette partie ne contient pas de résultats originaux. Les principales références utilisées concernant la théorie algébrique des codes sont dans [53] et dans [23].

1.1 Le codage

Voyons schématiquement le problème posé par la transmission à travers un canal bruité. Soit X,Y des alphabets. Considérons un mot $(u_1,\ldots,u_k) \in X^n$ qui représente l'information transmise.

$$\text{Information} \rightarrow (u_1,\ldots,u_k) \rightarrow \text{ Codeur } \rightarrow x = (x_1,\ldots,x_n) \in X^n,\ n > k$$

$$\rightarrow \text{ Canal bruité } \rightarrow y = (y_1,\ldots,y_n) \in Y^n \rightarrow \text{ Décodeur } \rightarrow (v_1,\ldots,v_k) \in X^k$$

Le Codeur transforme le mot émis en un mot du code utilisé. Dans le canal bruité, certains symboles peuvent être modifiés. On a alors une probabilité de transition

$$\mathbf{Prob}\left[y_i = y_i' \mid x_i = x_i'\right],$$

où la probabilité est prise sur l'alphabet Y, il s'agit de la probabilité qu'un élément reçu y_i soit égal à y_i' sachant que le symbole émis x_i est égal à x_i'.

1.1.1 Le canal binaire symétrique

On mesure la performance des codes dans certains modèles de canaux tels que le canal binaire symétrique. Ici on a $X = \mathbb{F}_2$, $Y = \mathbb{F}_2$ et dans ce contexte,

$$x_i \in X \longrightarrow \boxed{\text{canal}} \longrightarrow y_i \in Y,$$

et

$$\underset{Y}{\mathbf{Prob}}\,[y_i = y_i' \mid x_i = x_i']$$

représente alors la probabilité de transition. On note $p = \mathbf{Prob}\,[y_i = y_i' \mid x_i = x_i']$ si $y_i \neq x_i$ la probabilité qu'un bit soit mal transmis et donc $1 - p$ la probabilité qu'il soit bien transmis, si $y_i = x_i$. On peut représenter ceci par le schéma suivant :

$x_i \backslash y_i$	0	1
0	$1-p$	p
1	p	$1-p$

Il s'agit d'une transmission de symboles binaires, c'est pourquoi le canal est dit binaire. Le risque est le même de mal transmettre "0" ou "1" d'où le terme de canal symétrique. Chaque transmission est indépendante, le canal est dit sans mémoire. Avec ces hypothèses, propres au modèle, la proposition suivante est immédiate :

Proposition 1. *Soient p la probabilité de transmission incorrecte d'un bit. On suppose que la probabilité p est inférieure à 1/2. Si l'on transmet n bits sur un canal binaire symétrique, sans mémoire, la probabilité d'avoir j bits mal transmis est :*

$$\binom{n}{j} p^j (1-p)^{n-j}$$

L'introduction d'un code correcteur d'erreurs tend à améliorer cette situation.

1.1.2 Le canal adversaire

Dans un autre contexte, un modèle différent peut être étudié, c'est le modèle du "bruit adversaire". Dans ce modèle, on connaît juste la fraction de bits mal transmis, mais dans ce cas on suppose qu'un adversaire choisit judicieusement les bits qu'il va mal transmettre. Le canal est ici avec mémoire, c'est-à-dire que la probabilité que le i-ème bit soit bien transmis est dépendante de la bonne ou mauvaise transmission des bits émis précédemment et des bits qui vont être émis ultérieurement. En un mot, on ne fait aucune hypothèse sur la répartition des erreurs, tous les cas sont possibles, donc en particulier le canal binaire symétrique est inclus dans ce canal. On va rencontrer ce modèle, en étudiant certains systèmes cryptographiques, dans ce cas, le système est conçu pour simuler ce bruit adversaire. Ce modèle est aussi très pertinent sur le plan mathématique.

1.2 Les codes linéaires

L'alphabet considéré ici est un corps fini $\mathbb{F}$. Un code linéaire est d'abord un espace vectoriel sur le corps $\mathbb{F}$.

1.2.1 Quelques notations et définitions

Dans l'ensemble des messages de longueur n, on définit une métrique, la métrique de Hamming.

Définition 1. *Soit X un alphabet, $a \in X^n$ et $b \in X^n$. La distance de Hamming du mot a au mot b est :*

$$d(a,b) = |\{i \mid a_i \neq b_i\}|.$$

C'est une distance au sens classique du terme, [53]. On appellera distance de Hamming normalisée du mot a au mot b la quantité :

$$\Delta(a,b) = \frac{d(a,b)}{n}.$$

Quand n est grand, on interprète souvent la distance comme une probabilité :

$$\Delta(a,b) = 1 - \mathop{\mathbf{Prob}}_{i}[a_i = b_i],$$

où la probabilité est prise sur les indices $i \in [1,\ldots,n]$.

Définition 2. *Soit X un alphabet. Étant donné un mot $a \in X^n$ de longueur n, on définit le poids de $a \in X^n$ et on appelle poids de Hamming la quantité :*

$$wt(a) = |\{i \in [1,\ldots,n] \mid a_i \neq 0\}|.$$

Dans toute la suite de ce document, on notera $\mathbb{F}$ un corps fini. On notera $\mathbb{F}_p$ un corps premier du type $\mathbb{Z}/p\mathbb{Z}$ avec p premier. On notera $\mathbb{F}_q$, un corps du type $\mathbb{F}_{p^t}$, $t \geq 1$, avec p premier. On utilisera surtout les corps $\mathbb{F}_2$, et $\mathbb{F}_{2^t}$, $t > 1$. Donc, lorsqu'il n'y aura pas d'ambiguïtés, $\mathbb{F}_q$ désignera un corps du type $\mathbb{F}_{2^t}$, $t > 1$. Pour la construction de ces corps et leur implémentation nous renvoyons à [10], [9] et [53].

Définition 3. *Soit $\mathbb{F}$ un corps fini et soit $n > 0$. Le $\mathbb{F}$-espace vectoriel $\mathbb{F}^n$ est muni de la métrique de Hamming. Un code linéaire est un $\mathbb{F}$-sous-espace de $\mathbb{F}^n$. La distance minimale d d'un code linéaire C est égale au plus petit poids non nul de ce code :*

$$d = min\ \{d(a,b) \mid a \neq b,\ a,b \in C\} = \min_{a \in C \setminus \{0\}} \{wt(a)\}.$$

Soit E un ensemble d'erreurs possibles. Un code C corrige un nombre d'erreurs inférieur ou égal à t si

$$a + e \neq a' + e',\ \forall e,e' \in E \mid wt(e),wt(e') \leq t, \forall c \neq c' \in C.$$

Proposition 2. *Un code linéaire C corrige t erreurs ou moins si et seulement si la distance minimale de C satisfait*

$$d(C) \geq 2t + 1.$$

Remarque 1. *Pour sa définition stricte, les paramètres d'un code linéaire C sont: sa longueur n, sa dimension k, sa distance minimale d, i.e sa capacité de correction $\lfloor (d-1)/2 \rfloor$ et on parle d'un code $[n,k,d]$. Si le code n'est pas forcément linéaire on note (n,M,d), le code de longueur n et de paramètres M,d, où M est le cardinal du code et d la distance minimale de ce code.*

Définition 4. *Rappelons qu'un code linéaire C de longueur n sur l'alphabet $\mathbb{F}_2$ est un sous-espace vectoriel de $\mathbb{F}_2^n$. Ce sous-espace vectoriel admet un système de générateur $(g_i)_{1 \leq i \leq k}$ de cardinal, la dimension de ce sous-espace vectoriel, ce que l'on note habituellement k. La matrice $\mathcal{G}$ à k ligne et n colonne telle que $\mathcal{G}[i] = g_i$ est une matrice génératrice du code C. Cette matrice n'est bien sûr pas unique.*

1.3 Les codes cycliques

Lorsque l'on étudie les codes cycliques, il est judicieux de considérer les coordonnées des éléments qui composent les mots de code $0,1,\ldots,n-1$ comme des entiers modulo n.

Définition 5. *Un code linéaire C de longueur n sur un corps fini $\mathbb{F}$ est cyclique si et seulement si pour tout $a = (a_0,\ldots,a_{n-1}) \in C$:*

$$(a_0,\ldots,a_{n-1}) \in C \Longrightarrow (a_{n-1},a_0,\ldots,a_{n-2}) \in C.$$

Le vecteur $(a_{n-1},a_0,\ldots,a_{n-2})$ est obtenu à partir de a en effectuant le décalage $i \longmapsto i+1 \mod n$.

On va maintenant donner une définition plus constructive du code cyclique. Tout d'abord, on va identifier les mots $a = (a_0,\ldots,a_{n-1})$ d'un code cyclique C avec les polynômes $a(x) = a_0 + \ldots + a_{n-1}x^{n-1}$. Le fait que C soit invariant par le décalage s'exprime alors de la manière suivante:

$$a(x) \in C \Longrightarrow xc(X) \mod (x^n - 1) \in C.$$

On va donc considérer ces codes dans l'anneau principal

$$\mathcal{R}_n = \mathbb{F}[x]/(x^n - 1).$$

On peut donc maintenant donner une autre définition des codes cycliques:

Définition 6. *Un code cyclique C de longueur n sur $\mathbb{F}$ est un idéal principal de l'anneau $\mathcal{R}_n$.*

Rappelons quelques définitions avant de construire des codes cycliques. Par commodité, on se limite maintenant au corps $\mathbb{F}_q$.

Définition 7. *Soit α une racine primitive n-ième de l'unité dans une extension de $\mathbb{F}_q$. Cela veut dire que $1,\alpha,\ldots,\alpha^{n-1}$ sont tous différents et $\alpha^n = 1$. Pour chaque entier s avec $0 \leq s < n$, on note par $cl(s)$ la classe cyclotomique de $s \mod n$:*

$$cl(s) = \{s,qs,\ldots,q^{l-1}s \mod n\}$$

où l est le plus petit entier positif tel que n divise $q^l - 1$ ($\alpha \in \mathbb{F}_{q^l}$). Le polynôme minimal de α^s sur $\mathbb{F}_q$ est

$$M_{\alpha^s}(x) = \prod_{i \in cl(s)} (x - \alpha^i),$$

où les α_i, avec $i \in cl(s)$ sont appelés conjugués de α^s.

Remarque 2. *Si ω désigne une racine primitive du corps $\mathbb{F}_q$ alors on peut prendre $\alpha = \omega^{(q^l-1)/n}$. Notons que $M_{\alpha^s}(x)$ est un polynôme défini sur $\mathbb{F}_q$ car $\alpha \in \mathbb{F}_{q^l}$.*

Théorème 1. *[53] Soit C un code cyclique non nul de longueur n sur $\mathbb{F}_q$. Alors il existe un polynôme $g(x)$ appelé polynôme générateur de C ayant les propriétés suivantes :*

1. *$g(x)$ est l'unique polynôme monique de plus petit degré sur $\mathbb{F}_q$ engendrant l'idéal C dans $\mathcal{R}_n$:*

$$C = \{a(x)g(x) \mod (x^n - 1) \mid a(x) \in \mathbb{F}_q[x]\};$$

2. *$g(x)$ divise $x^n - 1$.*
3. *Soit $r = \deg(g)$, et soit $g(x) = \sum_{i=0}^{r}$ où $g_r = 1$. Alors la dimension de C est $k = n - r$; de plus, les polynômes*

$$g(x), xg(x), \ldots, x^{k-1}g(x)$$

forment une base de C. La matrice correspondante est donnée par :

$$\mathcal{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & & 0 \\ \vdots & & \ddots & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k} \end{bmatrix},$$

c'est une matrice génératrice du code C.

4. *Soit α une racine primitive n-ième de l'unité dans une extension de $\mathbb{F}_q$. Notons $M_{\alpha^s}(x)$ le polynôme minimal de α^s sur $\mathbb{F}_q$; alors*

$$g(x) = \prod_{s \in I} M_{\alpha^s}(x), \quad (g(\alpha^s) = 0),$$

où I est un sous-ensemble de représentants des classes cyclotomiques de q modulo n.

Exemple 1 :

$q = 2$, $n = 5$. Il est clair que $x^5 - 1$ divise $x^{15} - 1$, l'ordre multiplicatif de 2 modulo 5 est égal à 4 et donc $\mathbb{F}_{2^4}$ est le corps de décomposition de $x^5 - 1$. On détermine facilement les ω^k qui sont racines de $x^5 - 1$:

$$\begin{aligned} (\alpha^k)^5 = 1 &\iff 5k \equiv 0 \mod 15 \\ &\iff k \in \{0,3,6,9,12\}. \end{aligned}$$

Donc les racines de x^5-1 sont: $\{1,\omega^3,\omega^6,\omega^9,\omega^{12}\}$. Il n'y a que deux possibilités pour un polynôme générateur $g(x) \in \mathbb{F}_2[x]$:

$$g(x) = (x-1) \;\; ou \;\; g(x) = \frac{x^5-1}{x-1} = x^4+x^3+x^2+x+1.$$

Les classes de 2 modulo 5 sont:

$$\{0\}\ ,\ \{1,2,3,4\}.$$

◊

Définition 8. *Soit $q=2^t$ et $n=q^l-1$. Soit C le code cyclique de longueur n sur $\mathbb{F}_q$ et de polynôme générateur $g(x)$. Les racines de $g(x)$ sont appelées les zéros du code cyclique C. Donc le code cyclique est pleinement défini par ses zéros.*

1.4 Les codes BCH

Ces codes sont à l'origine de la définition générale des codes cycliques[1].

Définition 9. *Soit $q=2^t$ et $n=q^l-1$; notons α une racine primitive n-ième de l'unité dans $\mathbb{F}_{q^l}$. Soit h un entier tel que $2 \leq h \leq n$. Le code BCH de distance construite h est le code cyclique dont les zéros sont: $\alpha^s,\alpha^{s+2},...,\alpha^{s+h-1}$ et leurs conjugués. En d'autre termes, le polynôme générateur de ce code est*

$$g(x) = ppcm\{M_\alpha(x),M_{\alpha^2}(x),...,M_{\alpha^{h-1}}(x)\}$$

Remarque 3.

- *On considérera dans cette thèse uniquement le cas $q=2^t$, avec $t \geq 1$.*
- *On sait que la distance minimale d'un code BCH est supérieure ou égale à sa distance construite [23].*

Exemple 1. *Soit un code BCH de longueur 9 et de distance construite 4 sur $\mathbb{F}_2$.*

1. *Le corps de décomposition de X^9-1 est $\mathbb{F}_{2^6}$:*

$$x^9-1 \text{ divise } x^{63}-1,\ 63=2^6-1.$$

2. *Ici $s=0$ et $h=4$. On cherche $\alpha \in \mathbb{F}_{2^6}$ d'ordre 9.*

$$x^9-1=(x^3-1)(x^6+x^3+1) \Rightarrow \; \alpha^6+\alpha^3+1=0 \; \Rightarrow$$

$$\begin{aligned} x^9-1 = \; & (x-1)(x-\alpha^3)(x-\alpha^6) \\ & (x-\alpha)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)(x-\alpha^7)(x-\alpha^5) \end{aligned}$$

1. A.Hocquenghem *Codes Correcteur d'Erreurs*, Chiffres, 1959.

3. *On en déduit le polynôme générateur*

$$g(x) = M_{\alpha^0}(x)M_{\alpha^1}(x) = (x-1)(x^6+x^3+1) = x^7+x^6+x^4+x^3+x+1.$$

La dimension de ce code est $n-7=2$.

4. *La matrice génératrice montre que la distance minimale est 6 :*

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

Remarque 4. *On voit que dans l'exemple précédent la distance minimale peut être plus grande que la distance construite.*

1.5 Les codes cycliques de Reed-Solomon

On rappelle que l'on considère toujours le corps $\mathbb{F}_q$, avec $q = 2^t$. Le symbole ω désigne une racine primitive du corps $\mathbb{F}_q$. Les codes de Reed-Solomon sont une classe de codes cycliques définis sur l'alphabet $\mathbb{F}_q$.

Définition 10. *Un code de Reed-Solomon est un code BCH de longueur* $n = q-1$ *sur* $\mathbb{F}_q$.

Proposition 3. *Tous les codes de Reed-Solomon ont une distance minimale égale à leur distance construite.*

Preuve : Soit C un code BCH de longueur $q-1$ sur $\mathbb{F}_q$, de distance construite δ. Soit d sa distance minimale. Par définition, son polynôme générateur a la forme suivante :

$$g(X) = \prod_{i=1}^{\delta-1}(x-\omega^{l+i}).$$

On sait que $d \geq \delta$. Or il est clair que $0 < \omega(g) \leq \delta$. Donc

$$d = \delta \text{ avec } \delta = \deg g(X) + 1 = (q-1) - \dim C + 1.$$

Finalement : C est un code de longueur $q-1$, de dimension k et de distance minimale d, où $d = n-k+1$, et on le notera dans toute la suite $RS_q[q-1,k]$. □

Dans la suite on notera $\mathbb{F}_q^{(k)}[X]$ l'ensemble des polynômes de degré inférieur ou égal à k sur le corps $\mathbb{F}_q$.

Nous allons maintenant donner une interprétation polynômiale des codes de Reed-Solomon. Cette interprétation est nécessaire à la bonne compréhension des algorithmes de décodage qui existent et des testeurs que l'on évoquera plus tard.

Théorème 2. *[53] Soit* ω *un élément primitif de* $\mathbb{F}_q$ *et soit* k *un entier avec* $1 \leq k \leq q-1$. *Alors*

$$C = \{(p(1),p(\omega),p(\omega^2),\ldots,p(\omega^{q-2})) \mid p \in \mathbb{F}_q^{(k-1)}[x]\}$$

est le code de Reed-Solomon $RS_q[q-1,k]$.

Preuve : Le nombre de polynômes de degré inférieur à k est q^k, et comme un polynôme de $\mathbb{F}_q^{(k-1)}[X]$ à moins de $k-1$ racines où $k-1 \leq q-2$ racines, on en déduit que card $(C) = q^k$. Comme C est linéaire sur $\mathbb{F}_q$, C est un code linéaire de longueur $q-1$ et de dimension k. Soit C_1 un code de Reed-solomon $(RS_q[q-1,k])$. On va montrer que $C = C_1$. Comme ces deux codes sont de dimension k, il suffit de montrer que $C \subseteq C_1$. Soit $c(x) = c_0 + c_1 x + \ldots + c_{q-2}x^{q-2} \in C$ où $c_j = f(\omega^j)$, $(0 \leq j < q-1)$, pour des polynômes $f(x) = \sum_{l=0}^{k-1} f_l x^l \in \mathbb{F}_q^{(k-1)}[X]$. Rappelons que $c(x) \in C_1$. Si $c(\omega^i) = 0$ pour $1 \leq i \leq q-1-k$ alors

$$c(\omega^i) = \sum_{j=0}^{q-2} c_j \omega^{ij} = \sum_{j=0}^{q-2}(\sum_{s=0}^{k-1} f_s \omega^{js})\omega^{ij}$$

$$\sum_{s=0}^{k-1} f_s \sum_{j=0}^{q-2} \omega^{(i+s)j} = 0$$

parce que $\sum_{j=0}^{q-2} \omega^{(i+s)j} = \frac{\omega^{(i+s)(q-1)}-1}{\omega^{i+s}-1} = 0$ pour $1 \leq i+s \leq q-2$. Par suite $C \subseteq C_1$, et donc $C = C_1$. □

Enfin, voici une définition concernant les codes de Reed-Solomon raccourcis :

Définition 11. *On note* $RS_q[n,k]$ *le code de Reed-Solomon de longueur* $n < q$ *et de dimension* k. *C'est un code de Reed-Solomon dont on a omis certaines positions, c'est-à-dire, certains éléments de l'alphabet considéré ont été omis. Ainsi* n *peut être plus petit que la taille de l'alphabet sur lequel ce code est défini. On parle aussi du support du code* $RS_q[n,k]$, *ce sont les* n *éléments de* $\mathbb{F}_q$ *sur lesquels on considère les évaluations des polynômes de degré inférieur à* k.

1.6 Les codes de Reed-Muller

Les codes de Reed-Muller [53] sont une classe de codes cycliques.

Définition 12. *Soit* ω *une racine primitive de* $\mathbb{F}_{2^m}$. *Tout entier* $s \in [0,2^m-1]$ *peut être identifié par son développement 2-adique :*

$$s = \sum_{i=0}^{m-1} s_i 2^i, \quad s_i \in \{0,1\} \Longrightarrow s = (s_0,\ldots,s_{m-1}).$$

Le code cyclique de Reed-Muller de longueur 2^m-1 *et d'ordre* r, *noté* $RM[r,m]$, *est le code cyclique binaire dont les zéros sont*

$$S_r = \{\omega^s \mid 1 \leq wt(s) < m-r\}$$

où $wt(s)$ *est le poids de Hamming de* s.

Les codes de Reed-Muller peuvent être définis plus simplement en terme de fonctions booléennes.

Théorème 3. *Soient* m *un entier strictement positif,* $\{u_i \mid i \in [1,\ldots,2^m]\}$ *désigne l'ensemble des éléments de* $\mathbb{F}_2^m$. *Soit* $\mathbb{F}_2^{(r)}[x_1,\ldots,x_m]$ *pour* $r \geq 0$, *l'ensemble*

des polynômes multivariés de degré total au plus r et en m variables. Le code de Reed-Muller d'ordre r $(0 \leq r \leq m)$ *et de longueur* $n = 2^m - 1$ *est l'ensemble*

$$\{(f(u_1),\ldots,f(u_{2^m-1})) \mid f \in \mathbb{F}_2^{(r)}[x_1,\ldots,x_m]\}$$

Nous allons tout d'abord donner quelques rappels sur la construction des codes en général. Soient C_1, C_2 des codes de paramètres respectivement $(n1,M_1,d_1)$, $(n2,M_2,d_2)$. Leur somme directe consiste à former tous les vecteurs $|u|v|$ (i.e. la concaténation des vecteurs $u \in C_1$ et $v \in C_2$. C'est clairement un code de paramètres $(n_1 + n_2, M_1M_2, d = \min\{d_1,d_2\})$. Une construction plus fine est la construction de Plotkin :

Soient C_1, C_2 des codes de même longueur et de paramètres respectivement (n,M_1,d_1), (n,M_2,d_2). Alors, on peut former un nouveau code C_3, formé de tous les vecteurs

$$|u|u+v|,\ u \in C_1,\ v \in C_2.$$

Proposition 4. *Soient deux codes* C_1 *et* C_2 *ayant pour paramètres respectifs* (n,M_1,d_1) *et* (n,M_2,d_2). *Soit* C_3, *le code formé par l'ensemble des vecteurs*

$$|u|u+v|,\ u \in C_1,\ v \in C_2,$$

alors C_3 *est un code de paramètre* $(2n, M_1M_2, d = min\ \{2d_1,d_2\})$.

Théorème 4. *Construction de Plotkin pour les Reed-Muller :*

$$RM[r+1,m+1] = \{|u|u+v| \ :\ u \in RM[r+1,m],\ v \in RM[r,m]\}.$$

Donnons un équivalent en terme matriciel, soit $G(r,m)$ *une matrice génératrice de* $RM[r,m]$. *Alors,*

$$G(r+1,m+1) = \begin{pmatrix} G(r+1,m) & G(r+1,m) \\ 0 & G(r,m) \end{pmatrix}$$

est une matrice génératrice de $RM[r+1,m+1]$.

Preuve : Par définition un mot de code f de $RM[r+1,m+1]$ provient d'un polynôme $f(x_1,\ldots,x_{m+1})$ de degré au plus $r+1$, on peut alors écrire :

$$f(x_1,\ldots,x_{m+1}) = g(x_1,\ldots,x_m) + x_{m+1}h(x_1,\ldots,x_m),$$

où $\deg(g) \leq r+1$ et $\deg(h) \leq r$. Soient g, h les vecteurs de longueur 2^m correspondant respectivement à $g(x_1,\ldots,x_m)$ et à $h(x_1,\ldots,x_m)$. Bien sûr, $\mathbf{g} \in RM[r+1,m]$ et $\mathbf{h} \in RM[r,m]$. Maintenant considérons $g(x_1,\ldots,x_m)$ et $x_{m+1}h(x_1,\ldots,x_m)$ comme des polynômes en $x_1,\ldots,x_{m+1}$. Les vecteurs correspondants (de longueur 2^{m+1}) sont $|\mathbf{g}|\mathbf{g}|$ et $|0|\mathbf{h}|$. □

La conséquence immédiate est le théorème suivant :

Théorème 5. *Les codes de Reed-Muller* $RM[r,m]$ *ont une dimension égale à*

$$1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{r},$$

et ont une distance minimale égale à 2^{m-r}.

1.7 Décodage et problème de reconstruction

On a vu dans le premier chapitre que l'information était encodée avant de passer dans un canal. Pour coder, on utilise un code linéaire. Soit un code C de longueur n et de dimension k. Soit $\mathcal{G}$ la matrice génératrice de ce code. Avec une élimination de Gauss on peut prendre une matrice génératrice de la forme suivante, dite systématique :

$$\mathcal{G} = \begin{pmatrix} 1 & 0 & \dots & 0 & v_1^1 & \dots & v_1^{n-k} \\ & & & & & & \\ 0 & 0 & \dots & 1 & v_k^1 & \dots & v_k^{n-k} \end{pmatrix} = [I_k, M]$$

où I_k est la matrice identité $k \times k$ et M une matrice $k \times (n-k)$. Le codage se fait donc de la manière suivante :

$$\underset{\text{message}}{(x_1,\dots,x_k)} \begin{pmatrix} 1 & 0 & \dots & 0 & v_1^1 & \dots & v_1^{n-k} \\ & & & & & & \\ 0 & 0 & \dots & 1 & v_k^1 & \dots & v_k^{n-k} \end{pmatrix} = \underset{\text{mot émis}}{(x_1,\dots,x_k,y_1,\dots,y_{n-k})}$$

Le mot émis $a = (x_1,\dots,x_k,y_1,\dots,y_{n-k})$ est un élément du code C. Ce mot est ensuite transmis sur un canal bruité. C'est donc un mot de la forme $e + a$ que l'on va obtenir où e désigne l'erreur qui s'est ajoutée.

Définition 13. *Soit C un code de longueur n de dimension k et de distance minimale d ayant pour alphabet le corps $\mathbb{F}_q$. Soit $< .,. >$ le produit scalaire euclidien usuel : $\langle a,b \rangle = \sum_{i=1}^{n} a_i b_i$. On appel dual du code C son espace orthogonal :*

$$C^{\perp} = \{b \in \mathbb{F}_q^n \mid \forall a \in C : \langle a,b \rangle = 0\}.$$

On note $\mathcal{H}$ une matrice génératrice du code $C^{\perp}$ ainsi obtenue, elle est communément appelée matrice de parité du code C.

Remarque 5. *Il est clair que $C^{\perp}$ est un code de longueur n et de dimension $n-k$, mais il n'y a pas (a priori) de relation simple entre la distance minimale de C et celle de $C^{\perp}$. Si on choisit la matrice génératrice du code C de la forme $[I_k,M]$ où I_k est la matrice unité $k \times k$ et M une matrice $k \times (n-k)$, alors on voit aisément que $\mathcal{H} = [{}^tM, -I_{n-k}]$ est une matrice de parité du code $C^{\perp}$ où ${}^tM$ est la matrice transposée de la matrice M.*

Exemple 4 :

Le code de Hamming binaire de paramètres [7,4,3] admet pour matrice génératrice et pour matrice de parité :

$$\mathcal{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \mathcal{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

◊

C'est là que va commencer le décodage, mais avant tout il faut pouvoir détecter l'existence de l'erreur e du mot $a+e$ reçu. La capacité de détection du code C est $d-1$ puisque la distance minimale du code C est d, et la détection s'effectue ainsi :

$$a^t\mathcal{H} = 0 \Longleftrightarrow a \in C$$

où ${}^t\mathcal{H}$ est la transposée de la matrice de parité du code C et $a^t\mathcal{H}$ le syndrôme de a. En appliquant ceci à notre mot bruité on obtient

$$(a+e)^t\mathcal{H} = e^t\mathcal{H}.$$

On prouve alors facilement que :

- tous les mots du translaté $e+C$ ont le même syndrôme;
- si $b^t\mathcal{H} = e^t\mathcal{H}$, alors $b \in e+C$;

donc si le poids de e vérifie $\omega(e) \leq \frac{d-1}{2}$, sachant que $\omega(a) \geq d$ pour tout $a \in C^*$, on obtient :

$$\omega(e) = \min\{\omega(e+b) \mid b \in C\}.$$

Le vecteur e est l'unique mot de poids minimum d'un translaté de C. Si $\omega(e) > \frac{d-1}{2}$ il est possible que plusieurs mots de C soient à une même distance du mot reçu $e+a$.

Cette résolution formelle du décodage induit l'algorithme naturel de décodage par table, inutilisable dans la pratique lorsque la table devient trop grande. Il est donc nécessaire de développer des algorithme de décodage efficaces.

Il faut noter qu'il peut être utile de savoir résoudre ce problème de décodage au-delà de la distance minimale. Dans ce cas, plusieurs mots du code considéré peuvent être équidistants du mot reçu. Il existe deux types de décodage que nous allons définir :

Définition 14. *Soit C un code sur $\mathbb{F}_q$. Soit $a \in C$ un mot émis à travers le canal q-symétrique et c le mot reçu.*

1. *On appelle décodage à maximum de vraisemblance, l'obtention du mot de code qui a la plus grande probabilité d'être le plus proche du mot reçu c.*
2. *On appelle décodage par liste, l'obtention de tous les mots du code C les plus proches du mot reçu c, c'est-à-dire tels que la distance de Hamming qu'il y a entre ce mot reçu et les mots du code C les plus proches soit inférieure à un seuil que l'on s'est fixé.*

Ce problème de décodage est classique au sens des télécommunications. Mais nous allons maintenant aborder un problème particulier de décodage.

1.7.1 Décodage en grandes longueurs

Supposons que la longueur du code soit si grande que le simple stockage de l'un de ses mots soit impossible. Cette considération a un sens lorsque l'on considère les codes de Reed-Muller et Reed-Solomon puisque l'on a vu précédemment qu'un mot de ces codes peut tout simplement être représentés par un polynôme. Il faut cependant que le degré de ces polynômes soit raisonnable, i.e. que la dimension du code soit petite.

Conditions du décodage

On suppose toujours que le mot de code est transmis par un canal bruité. Simplement, on suppose maintenant que l'on a accès à une simple fraction des positions qui composent ce mot de code. Il revient au même de dire, qu'on a accès seulement à une fraction des images du polynôme qui correspond au mot de code bruité. Dans ce contexte, on va s'intéresser au problème suivant :

Définition 15. *(Problème de reconstruction). Soient un corps $\mathbb{F}_q$, n couples d'éléments $\{(x_i,y_i)\}_{i=1}^n$ de $\mathbb{F}_q^m \times \mathbb{F}_q$ pour $m \geq 1$, avec y_i distincts et les entiers e et h. Alors on appelle reconstruction l'action qui consiste à construire tous les polynômes $f : \mathbb{F}_q^m \to \mathbb{F}_q$ satisfaisant :*

le polynôme $f(x)$ est de degré au plus h et $card\{i \mid f(x_i) \neq y_i\} \leq e$. Ce type particulier de décodage est en fait un problème d'interpolation avec erreur.

Remarque 6. *Lorsque le polynôme considéré est univarié on note k son degré. Si le polynôme considéré est multivarié on note r le degré total de ce polynôme. Le problème de reconstruction s'énonce de la même manière dans le cadre univarié, c'est-à-dire encore dans le contexte des codes de Reed-Solomon, que dans le cadre multivarié, c'est-à-dire encore dans le contexte des Reed-Muller. Dans la définition précédente on a réuni les deux contextes en notant h le degré ou degré total suivant le cas considéré.*

Définition 16. *On parlera de clairs choisis lorsque l'on supposera que le programme correcteur peut choisir la fraction des positions d'un mot de code, ou de manière équivalente, lorsqu'il peut choisir les entrées de la fonction qui correspond directement à ce mot de code. On parlera de clairs connus lorsque la fraction des positions auxquelles le programme correcteur a accès nous est donnée.*

Pour les codes de Reed-Muller

Le problème de reconstruction dans le contexte multivarié est considéré comme un problème difficile. Certains résultats de complexité ont été introduits en 1995 par Goldreich, Sudan et Rubinfeld [16].

Il existe diverses variantes du problème de reconstruction. Certaines hypothèses sont ajoutées afin de rendre le problème plus accessible. Par exemple, on

peut supposer que la fraction des positions :

$$\{(x_i,y_i)\}_{i=1}^{n}$$

est choisie par le programme correcteur. Certaines hypothèses peuvent parfois complexifier le problème, comme par exemple le choix du canal par lequel le mot de code a été transmis. Nous reviendrons ultérieurement sur ces considérations. En particulier, on étudiera des algorithmes de reconstruction qui sont de complexité polynomiale en le logarithme de la longueur du code considéré.

En 1995, Goldreich, Sudan et Rubinfeld [16] ont introduit un nouveau type d'algorithme qui répond au problème de reconstruction dans le cas où l'on choisit les entrées de la fonction que l'on étudie. Ce type de décodage présente plusieurs avantages :

1. Il s'agit de décodage par liste, ce qui est plus fort que le décodage à maximum de vraissemblance puisque l'on obtient la liste des mots de code les plus proches contre le mot de code le plus proche.
2. La complexité de ce type d'algorithme est très intéressante puisqu'il ne s'agit plus de complexité en $\mathcal{O}(n)$ mais $\mathcal{O}(poly(\log(n)))$ où n est la longueur du code considéré.

Pour les codes de Reed-Solomon

Dans le contexte univarié, le problème n'est pas moins simple, cependant un algorithme très efficace permet de répondre à ce problème, mais avec tout de même des hypothèses par rapport au problème de reconstruction classique. Cet algorithme a été introduit en 1997 par M. Sudan [45], puis amélioré en 1998 par M. Sudan et V. Guruswami [17]. Cet algorithme est communément qualifié d'algorithme de décodage par liste, il retourne la liste des polynômes répondant au problème posé :

Théorème 6. *[17] Soit $f : \mathbb{F}_q \to \mathbb{F}_q$. Soient n le nombre de couples $(x_i,f(x_i))$ qui sont donnés. Soit k le degré des polynômes p que l'on veut reconstruire. Soit $e = \epsilon(\tau)n$ le nombre maximal d'erreurs, c'est-à-dire le nombre maximal de couples $\{(x_i,f(x_i))\}$ pour lesquels $f(x_i) \neq p(x_i)$. Alors si*

$$e < n - \sqrt{kn},$$

on peut reconstruire les polynômes p qui vérifient $|\{i \mid f(x_i) = p(x_i)\}| > n-e$. La complexité de l'algorithme permettant cette reconstruction est en $\mathcal{O}(n^2\log^2(n))$ opérations arithmétiques sur le corps $\mathbb{F}_q$.

Remarque 7. *Reprenons les hypothèses du théorème précédent. Cet algorithme est très intéressant, car il montre que si n est très grand, une fraction des évaluations de la fonction f, prise sur des éléments aléatoires de $\mathbb{F}_q$, permet de reconstruire les polynômes de degré au plus $k-1$ qui coïncident le plus largement avec*

f, il suffit simplement que la condition $e < n - \sqrt{kn}$ soit vérifiée. Supposons que la distance normalisée de fonction f au code de Reed-Solomon soit égale à $1 - \mu$, c'est-à-dire qu'il existe au moins un polynôme $P \in \mathbb{F}_q[X]$ tel que P coïncide avec f sur une fraction au moins égale à μ des entrées. Un simple résultat d'échantillonage (cf chapitre 4) permet de dire qu'il faut $\mathcal{O}(\mu^{-2})$ couples d'entrée sorties de la fonction f pour approximer cette propriété statistique. Avec les notations du théorème précédent, cela implique que

$$\mu = \frac{n-e}{n} > \frac{\sqrt{k}}{\sqrt{n}},$$

d'où $n \approx k\mu^{-2}$. Nous reviendrons ultérieurement sur ce théorème.

On a abordé ici le problème de reconstruction. On s'intéressera par la suite à un problème très proche, nous n'essaierons pas de construire directement les polynômes qui répondent au problème de reconstruction, mais nous chercherons d'abord à savoir s'ils existent. C'est la problématique des testeurs.

Chapitre 2

Le chiffrement par bloc

Nous allons simplement replacer le principe du chiffrement dans son contexte, et introduire brièvement différents types de systèmes existants. Pour faire cette brève introduction au chiffrement, on s'est inspiré de différents articles de vulgarisation figurant sur Internet[1].

2.1 Concepts généraux

La cryptographie est vielle de plus de 2000 ans et bien sûr cette science a beaucoup évolué depuis ses origines, surtout depuis l'apparition des traitements automatisés de l'information, mais les principes anciens restent encore d'actualité aujourd'hui.

Le chiffrement est la transformation d'une information intelligible (un texte de départ par exemple) en une information qui ne pourra pas être comprise par des personnes qui ne seraient pas autorisées à lire ces informations. C'est l'idée de base. Aussi, un des objectifs fondamentaux de la cryptographie est de permettre à deux personnes, appelées traditionnellement Alice et Bob de communiquer par l'intermédiaire d'un canal de transmission public (une ligne de téléphone ou un réseau par exemple tous deux réputés peu sûrs), sans qu'un espion éventuel appelé Oscar en comprenne le sens. Alice transforme le message de départ x par un procédé de chiffrement (ou codage) noté F en un message y, et l'envoie alors à Bob. Oscar qui espionne le canal de transmission ne peut pas comprendre le message car il ne connaît pas la façon de procéder pour le déchiffrer (ou le décoder). Ce n'est pas le cas de Bob qui peut appliquer le procédé inverse de celui d'Alice, et transformer le message chiffré y pour qu'il soit identique au message x d'origine.

Le message est couramment appelé texte clair. Le processus de transformation d'un message intelligible en un message incompréhensible est appelé chiffrement.

1. http://www.chez.com/nopb/crypto1.html

Le résultat de ce processus de chiffrement est appelé texte chiffré. Le processus de reconstruction du texte clair à partir du texte chiffré est appelé déchiffrement. On a représenté ces différents processus dans la figure 2.1

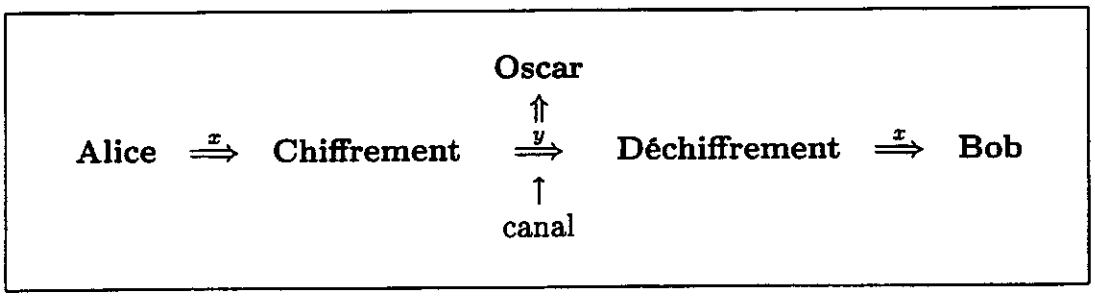


FIG. 2.1 – *Système de chiffrement sur un canal de transmission public*

2.2 Pourquoi la cryptographie, le chiffrement ?

Avant l'utilisation massive des ordinateurs, les principaux utilisateurs moyens de chiffrement furent les gouvernements et les militaires (pour échanger des ordres ou des informations secrètes par exemple). Pour les utilisateurs le besoin de conserver le secret de leur communications était primordial. Aujourd'hui le chiffrement est toujours une arme importante pour les militaires, mais elle est également utilisée par les banques, les institutions financières et les industries. Le chiffrement a deux problèmes majeurs à résoudre. Le premier est la protection des fichiers de données enregistrés sur des supports de masse (CD-rom par exemple) ou échangées sur le réseau contre des destructions ou des modifications non voulues. C'est ce que l'on nomme le problème d'intégrité des données : des modifications de données non autorisées par un tiers doivent être détectées. Le second est le problème de la confidentialité. Protéger les informations importantes d'une entreprise devient de plus en plus essentiel pour maintenir un niveau de compétitivité suffisant. Le vol ou la destruction de données confidentielles entraîne pour une entreprise des dommages financiers très importants. La cryptographie permet souvent en outre l'authentification, c'est-à-dire de s'assurer de l'identité d'une personne, que celle-ci est bien celle à qui l'on s'adresse sans aucun doute possible. Enfin la cryptographie (la signature) peut garantir l'envoi d'une information et sa réception sans que l'émetteur, ni le récepteur puissent nier la transaction ainsi que son contenu (très important pour le commerce électronique), on parle alors de droit de non répudiation.

2.3 Principales méthodes existantes

Avant l'apparition des ordinateurs, la sécurité du chiffrement reposait sur le secret des opérations réalisées (des méthodes de substitution ou de transposition en majorité), il suffisait de connaître la façon de coder, pour pouvoir décoder très facilement (on parlera de chiffrement restreint). Maintenant ces méthodes n'ont plus qu'un intérêt historique, car dans la pratique elles sont très vite inutilisables (du point de vue de leur sécurité). Aujourd'hui, les nouveaux algorithmes de chiffrement utilisés sont publics, et leur sécurité repose sur le concept des clefs.

On distingue deux classes d'algorithmes à base de clefs : les premiers sont dits symétriques et les seconds asymétriques (cf. figure 2.2 et 2.3). La différence est que les algorithmes symétriques utilisent la même clef (ou alors est facilement dérivée de celle-ci) pour chiffrer ou déchiffrer, alors que les seconds utilisent une clef de déchiffrement différente de la clef de chiffrement (et qui ne peut être non plus facilement dérivée de celle-ci).

Dans le premier cas, l'émetteur (Alice) et le destinataire (Bob) doivent se mettre d'accord sur une clef à utiliser avant d'échanger des messages. Cette clef doit rester secrète. La sécurité d'un algorithme symétrique repose sur la clef : si celle-ci est dévoilée, alors n'importe qui peut déchiffrer les messages d'Alice.

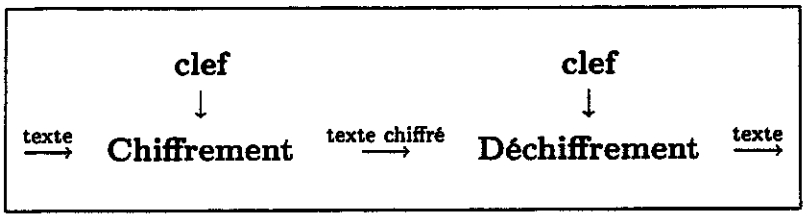


FIG. 2.2 – *Chiffrement symétrique*

Quant aux algorithmes asymétriques, ils permettent à la différence des premiers que la clef de chiffrement soit rendue publique, c'est pourquoi on appelle souvent la clef de chiffrement : clef publique, et la clef de déchiffrement clef privée. N'importe qui peut utiliser la clef de chiffrement pour chiffrer un message mais seul celui qui possède la clef de déchiffrement peut déchiffrer le message chiffré résultant.

2.4 Le chiffrement moderne

Les algorithmes de chiffrement étaient peu sûrs en général par le passé, le cassage du système FANTASIA pendant la seconde guerre mondiale reposant à la fois sur des transpositions et des substitutions combinées atteste de la vulnérabilité de ces techniques.

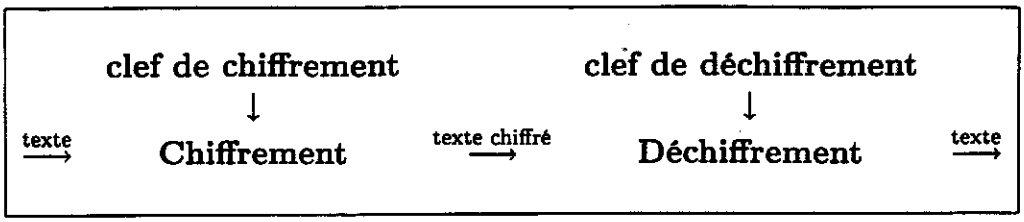


FIG. 2.3 – *Chiffrement asymétrique*

Le chiffrement moderne utilise la puissance des ordinateurs. Comme les données traitées par les ordinateurs sont uniquement sous forme numériques (bits), les procédés de substitution et de transposition sont toujours utilisés. Ce changement de dimension rend plus sûr les techniques de chiffrement actuelles. Comme nous l'avons déjà rapidement vu, on distingue deux types d'algorithmes à clefs : les systèmes de chiffrement symétriques et les systèmes asymétriques.

Les problèmes des systèmes symétriques sont les suivants :

1. Si la clef secrète est compromise par un opposant, alors ce dernier peut déchiffrer tous les messages codés avec celle-ci. Oscar peut même se faire passer pour Alice ou Bob.
2. Les clefs doivent être distribuées secrètement : c'est difficile à l'échelle planétaire.
3. Si une clef différente est utilisée pour chaque paire différente d'utilisateur du réseau, le nombre total des clefs augmente très rapidement en fonction du nombre total d'utilisateurs.

Un système qui était utilisé il y a encore peu de temps était le DES (Data Encryption Standard).

2.4.1 Le DES

Le DES est resté un standard pendant plus de 15 ans. Il a été développé en 1976 par IBM pour le N.B.S.(National Bureau of Standards). Bien qu'il montre maintenant des signes de vieillesse, il a remarquablement bien résisté à des années de tentatives cryptanalyse.

C'est un algorithme à clef secrète qui chiffre un bloc de texte clair de 64 bits en utilisant une clef de 56 bits. Il utilise les deux grandes lois de Shanon : diffusion (en utilisant des permutations) et confusion (en utilisant des substitutions) de bits pour casser la fréquence d'apparition des lettres dans le texte clair, et compliquer le lien entre le fichier codé et la clef secrète utilisée. Sa résistance aux attaques possibles (déjà connues) est bonne. C'est un système particulier dit système de chiffrement symétrique itératif par blocs.

Dans la suite de ce document, on s'intéressera à cette classe particulière de chiffrements symétriques, en particulier on donnera une description du DES.

2.5 Le chiffrement symétrique itératif par blocs

2.5.1 Le chiffrement par blocs

Principe

Un chiffrement est dit par blocs s'il divise le texte clair en blocs de taille fixe (généralement 64 ou 128 bits) et chiffre un bloc à la fois avec la même clé secrète. La méthode la plus simple pour chiffrer un message dont la longueur dépasse la taille d'un bloc, consiste à diviser le message en plusieurs blocs et à chiffrer chacun des blocs séparément. Ce mode opératoire appelé ECB (Electronic Codebook) présente de nombreux inconvénients, en particulier, deux blocs de clair identiques produisent deux blocs chiffrés identiques, ce qui permet à un attaquant de détecter d'éventuelles répétitions dans le texte clair. Dans la pratique, les chiffrements par blocs sont utilisés avec d'autres modes opératoires plus sûrs dans lesquels chaque bloc de chiffré dépend de tous les blocs de clair précédents. On a essentiellement trois modes opératoires, les modes CBC (Cipher Block Chaining), OFB (Output Feedback) et CFB (Cipher Feedback). On décrit à la figure 2.4, le mode CBC. Sur ce schéma, IV est un vecteur d'initialisation constant.

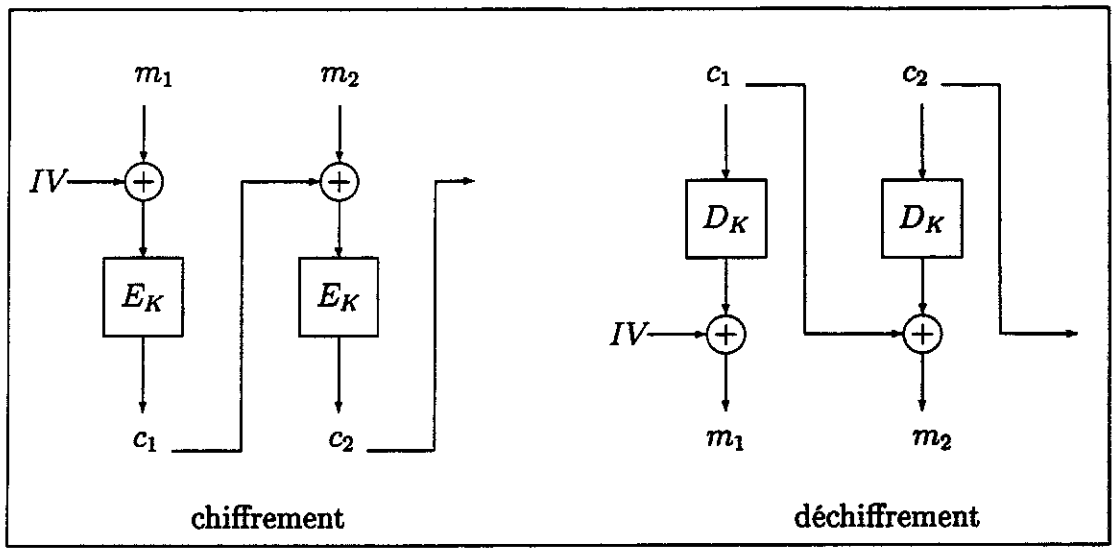


FIG. 2.4 – *Le mode opératoire CBC*

2.5.2 Recherche exhaustive de la clé

Un paramètre essentiel pour la sécurité d'un système à clé secrète est la taille de l'espace des clés, c'est-à-dire le nombre de clés possibles. En effet, il est toujours possible d'imaginer sur un algorithme de chiffrement une attaque dite *exhaustive* pour retrouver la clé. L'attaque consiste simplement à énumérer l'ensemble des clés possibles et de les essayer successivement pour déchiffrer un message. On considère la clé d'un système de chiffrement symétrique comme un mot de k bits, le nombre moyen de clés à fournir à la fonction de déchiffrement pour mener à bien cette attaque est de 2^{k-1}. La faisabilité d'une telle attaque dépend évidemment de l'évolution de la technologie. On considère actuellement que l'espace des clés est suffisamment grand si la clé comporte au minimum 80 bits, c'est pourquoi les algorithmes actuels proposent en général au minimum des clés de 128 bits. Ainsi, l'algorithme de chiffrement à clé secrète le plus utilisé jusqu'à très récemment, le DES (Data Encryption Standard), est désormais vulnérable à une attaque exhaustive puisqu'il utilise une clé secrète de 56 bits. Une telle attaque, demandant en moyenne 2^{55} chiffrements DES, a été réalisée en janvier 1998 en 39 jours sur 10 000 Pentium en parallèle, puis en 22 heures en juillet 1998 à l'aide d'une machine dédiée (EFF DES Cracker) comportant 1500 composants DES[2]. Le coût d'une telle machine est estimé à 250000 Dollars. C'est pourquoi le DES a récemment été remplacé par un nouveau standard de chiffrement à clé secrète, l'AES (Advanced Encryption Standard)[27]. L'AES a été choisi en octobre 2001 parmi les 15 systèmes proposés en réponse à l'appel d'offre lancé par le NIST (National Institute of Standards and Technology). Cet algorithme, initialement appelé RIJNDAEL, a été conçu par deux chercheurs belges, V. Rijmen et J. Daemen [27]. Il opère sur des blocs de message de 128 bits et est disponible pour trois tailles de clé différentes, 128, 192 et 256 bits, ce qui le met à l'abri des attaques exhaustives.

Toutefois, il existe d'autres types d'attaques sur ces systèmes. La plupart sont des attaques de type "sur la dernière clé". C'est ce dernier type d'attaques qui va être détaillé par la suite.

On considère qu'un chiffrement à clé secrète présente une bonne sécurité s'il n'existe pas d'attaque dont la complexité soit significativement inférieure à la recherche exhaustive. A l'heure actuelle, la sécurité des systèmes à clés secrètes repose uniquement sur la constatation qu'ils sont difficiles à cryptanalyser. On sait démontrer qu'un algorithme résiste aux attaques *classiques*, mais on ne peut garantir sa résistance contre de nouveaux types d'attaques.

2. http://www.eff.org/descracker.html

2.5.3 Le chiffrement itératif

L'idée générale d'un chiffrement itératif est de construire un algorithme à l'aide de plusieurs unités réalisant des opérations élémentaires de chiffrement, de manière à ce que le chiffrement résultant soit cryptographiquement plus résistant que chacun des composants pris isolément. Cette technique a été formalisée par Horst Feistel au début des années 70, alors qu'il travaillait chez IBM sur l'algorithme LUCIFER qui devait servir de base à la conception du DES. Les notions fondamentales utilisées sont tirées de l'article fondateur de Claude Shannon, *"The communication theory of secrecy systems"*[41], où sont traitées les bases mathématiques d'un système de communication chiffrée, à partir de la théorie de l'information. Ont été dégagées en particulier les notions de diffusion et de confusion. La confusion permet de rendre inextricables les liens entre le message en clair, la clé et le message chiffré. La diffusion permet de réduire les possibilités d'utilisation des données statistiques présentes dans le texte en clair en diluant ses données fréquentielles tout au long du texte chiffré.

Cette section utilise les notes de cours de Anne Canteaut: "Cryptanalyse différentielle et cryptanalyse linéaire des chiffrements de type DES" [1].

Principe général

De manière plus formelle, un chiffrement itératif par blocs consiste à itérer r fois une fonction interne F (cf. Figure 2.5). A chacun des r tours, la fonction F est paramétrée par une quantité secrète K_i, la clé du tour. Pour que le chiffrement soit inversible, la fonction itérée F doit être une permutation pour chaque valeur possible du paramètre K_i. Les r clés $(K_1, \cdots, K_r)$ sont en général dérivées d'une unique clé-maître par un algorithme de cadencement de clé.

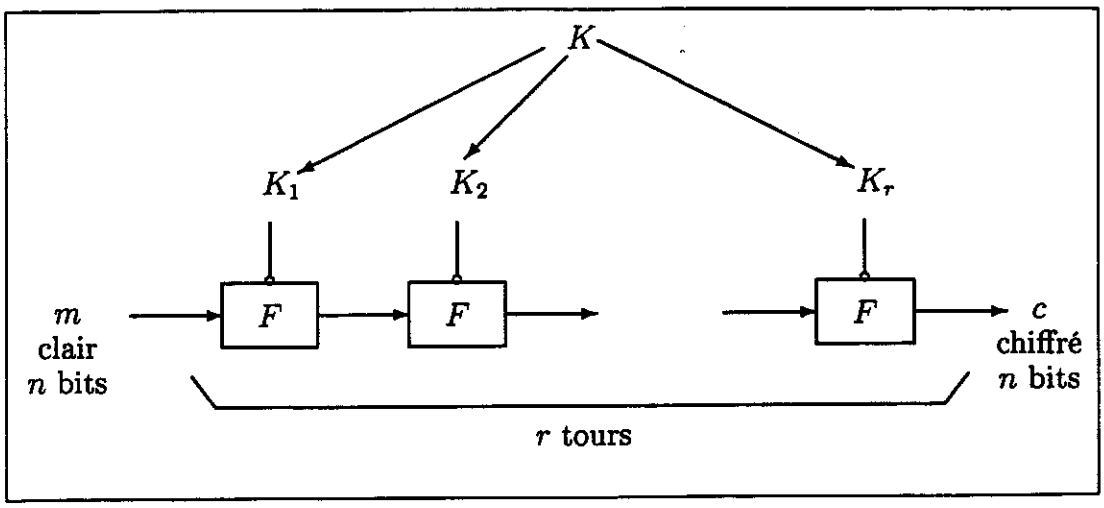


FIG. 2.5 – *Schéma d'un chiffrement itératif par blocs*

La plupart des algorithmes par blocs sont construits suivant ce modèle (DES, AES, IDEA, SAFER ...). Dans le DES, les blocs comportent 64 bits. La fonction itérée F est une permutation de $\mathbf{F}_2^{64}$ paramétrée par une clé de 48 bits. Cette fonction est itérée 16 fois et les 16 clés $(K_1, \cdots, K_{16})$ sont dérivées d'une unique clé de 56 bits. Quant à l'AES, pour une clé de 128 bits, il itère 10 fois une permutation de $\mathbf{F}_2^{128}$ paramétrée par une sous-clé de 128 bits.

Les algorithmes itératifs par blocs se répartissent essentiellement en deux grandes familles, suivant la structure de la fonction interne F. Les deux structures principalement utilisées sont la structure de Feistel (utilisée dans le DES), que nous étudions plus en détail par la suite, et la structure substitution-permutation (utilisée dans l'AES).

2.5.4 Chiffrement de type Feistel

Dans la suite du document, lorsqu'on considère des mots binaires dans un espace vectoriel, on note + le "ou exclusif" bit-à-bit.

Dans un chiffrement de Feistel, la fonction interne F possède la structure suivante :

Définition 17. Chiffrement de Feistel
Un chiffrement de Feistel est un chiffrement itératif par blocs opérant sur des blocs de $2n$ bits. La fonction itérée F est définie par :

$$\begin{array}{rccc} F: & \mathbf{F}_2^n \times \mathbf{F}_2^n & \to & \mathbf{F}_2^n \times \mathbf{F}_2^n \\ & (L_{i-1}, R_{i-1}) & \mapsto & (L_i, R_i) \end{array}$$

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

Quelle que soit la fonction de tour f utilisée, un chiffrement de Feistel est inversible. Pour déchiffrer, il suffit d'utiliser le même processus à r tours en

inversant l'ordre des clés $K_1, \cdots, K_r$.

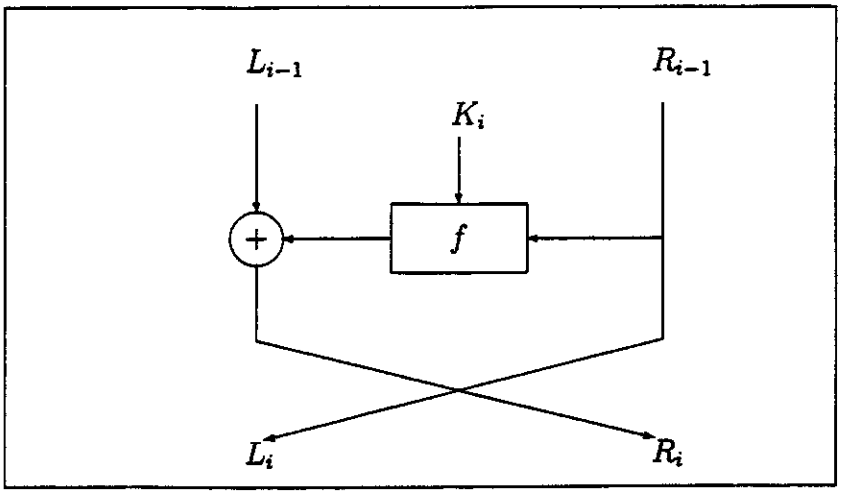


FIG. 2.6 – *Fonction itérée d'un chiffrement de Feistel*

Le DES

Dans le cas particulier du DES, la fonction itérée possède la forme décrite dans la figure 2.7. Conformément au standard, on numérote à partir de la gauche les bits d'un mot $x \in \mathbf{F}_2^n$, transitant dans le DES : $x = (x_1, \cdots, x_n)$. On transforme tout d'abord les 32 bits de la partie droite de l'entrée, R_{i-1}, en un mot de 48 bits par une expansion affine E. On lui ajoute ensuite les 48 bits de la clé K_i par un ou exclusif et on prend l'image du résultat suivant une fonction $S : \mathbf{F}_2^{48} \to \mathbf{F}_2^{32}$. Puis les 32 bits de la sortie de S sont permutés par une permutation P.

$$\begin{array}{rccc} f: & \mathbf{F}_2^{32} \times \mathbf{F}_2^{48} & \to & \mathbf{F}_2^{32} \\ & (R_{i-1}, K_i) & \mapsto & P(S(E(R_{i-1}) + K_i)) \end{array}$$

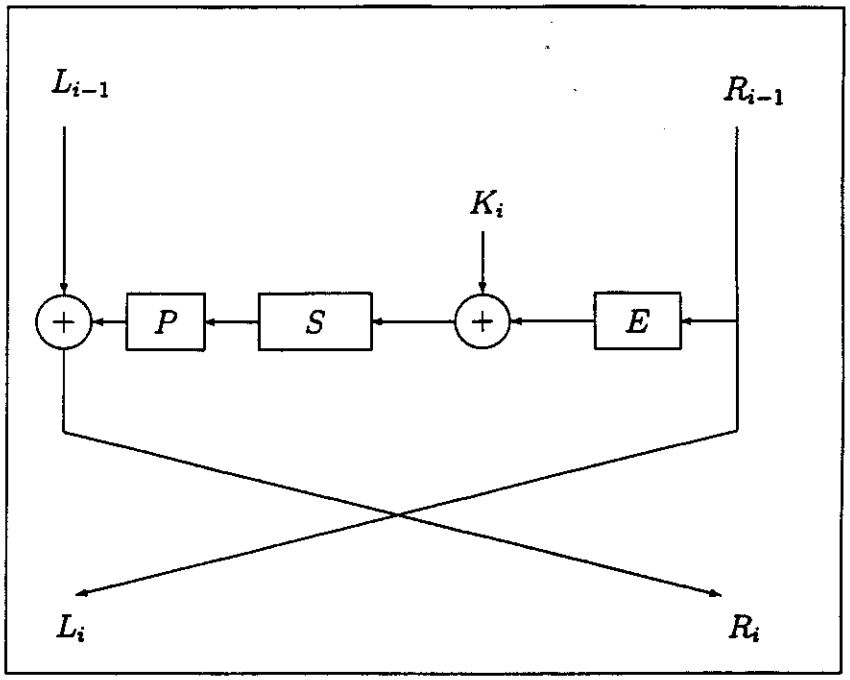


FIG. 2.7 – *Fonction itérée du DES*

L'expansion affine E est définie par

$$\begin{array}{rccc} E: & \mathbf{F}_2^{32} & \to & \mathbf{F}_2^{48} \\ & (x_1 \cdots x_{32}) & \mapsto & (y_1 \cdots y_{48}) = (x_{32}x_1 \cdots x_{31}x_{32}x_1) \end{array}$$

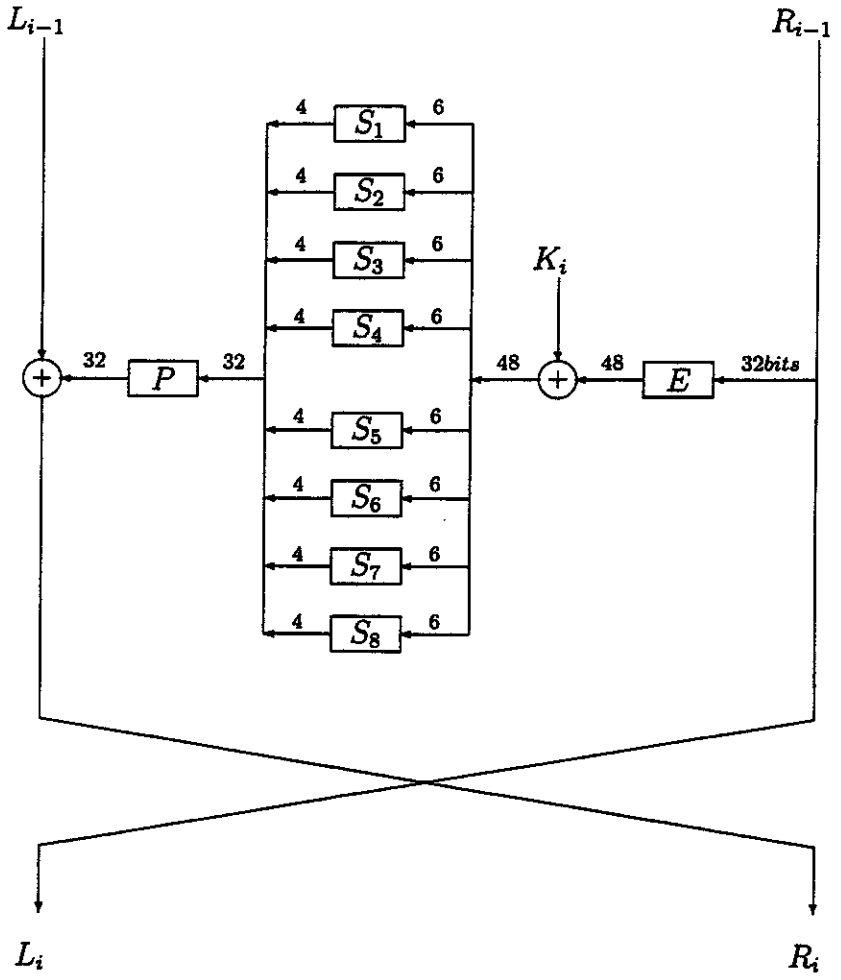
Les bits de y correspondent aux bits de x pris dans l'ordre suivant :

E					
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

La permutation P transforme $(x_1 \cdots x_{32})$ en $(y_1 \cdots y_{32}) = (x_{16}x_7 \cdots x_4x_{25})$ où l'ordre des bits de y est donné par :

P							
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

La fonction S, quant à elle, est définie par les 8 *boîtes-S* du DES. Le mot de 48 bits $E(R_{i-1}) + K_i$ est divisé en 8 blocs de 6 bits $(B_1, \cdots, B_8)$. Sur chacun de ces blocs B_i opère la fonction $S_i : \mathbf{F}_2^6 \to \mathbf{F}_2^4$. Les fonctions E et P étant affines, c'est la fonction S qui réalise la confusion (au sens de Shannon).



Le schéma général du DES est détaillé dans la figure 2.8. Notons qu'avant le premier tour, on permute les 64 bits du clair par la permutation IP. La permutation inverse IP^{-1} est appliquée sur les 64 bits obtenus à la fin des 16 itérations. On ne tiendra pas compte dans la suite de ces deux permutations qui n'influent pas sur la sécurité du système.

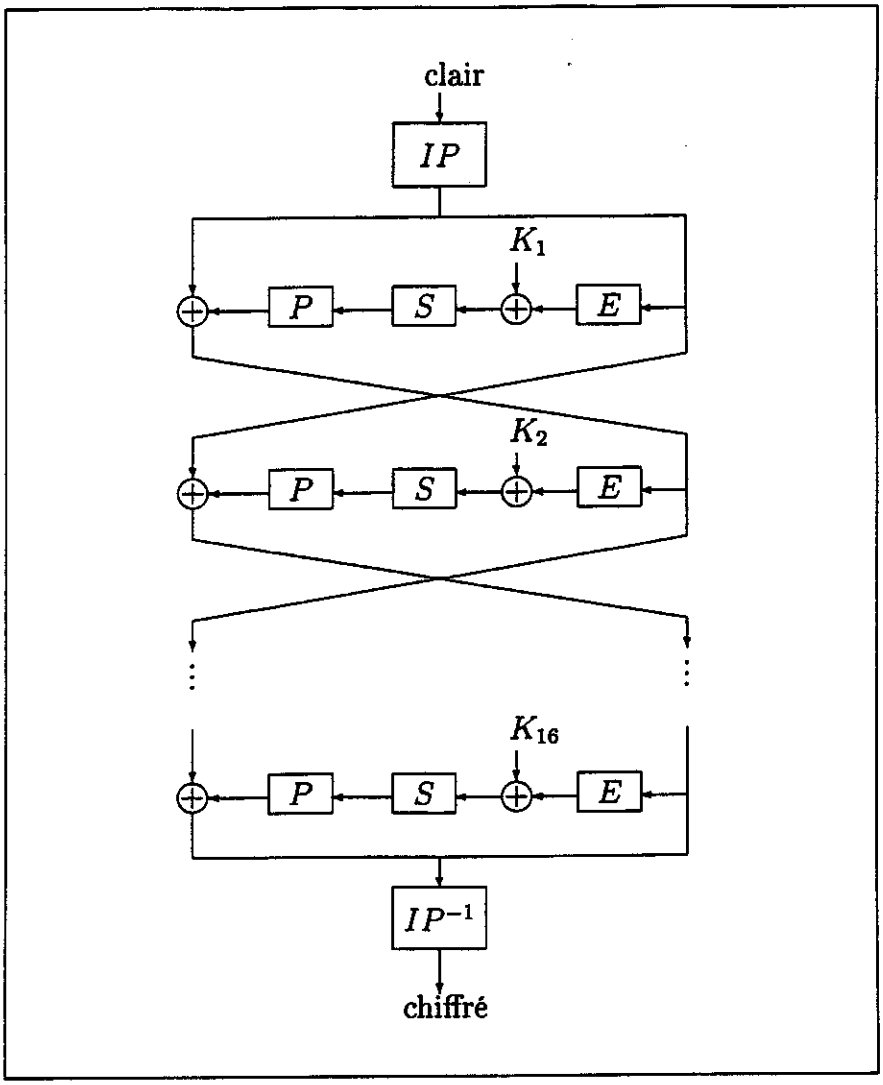


FIG. 2.8 – *Le DES*

On donne par ailleurs le schéma de cadencement de clé du DES à la figure 2.9. Le DES utilise 16 sous-clés de 48 bits dérivées d'une clé-maître de 56 bits.

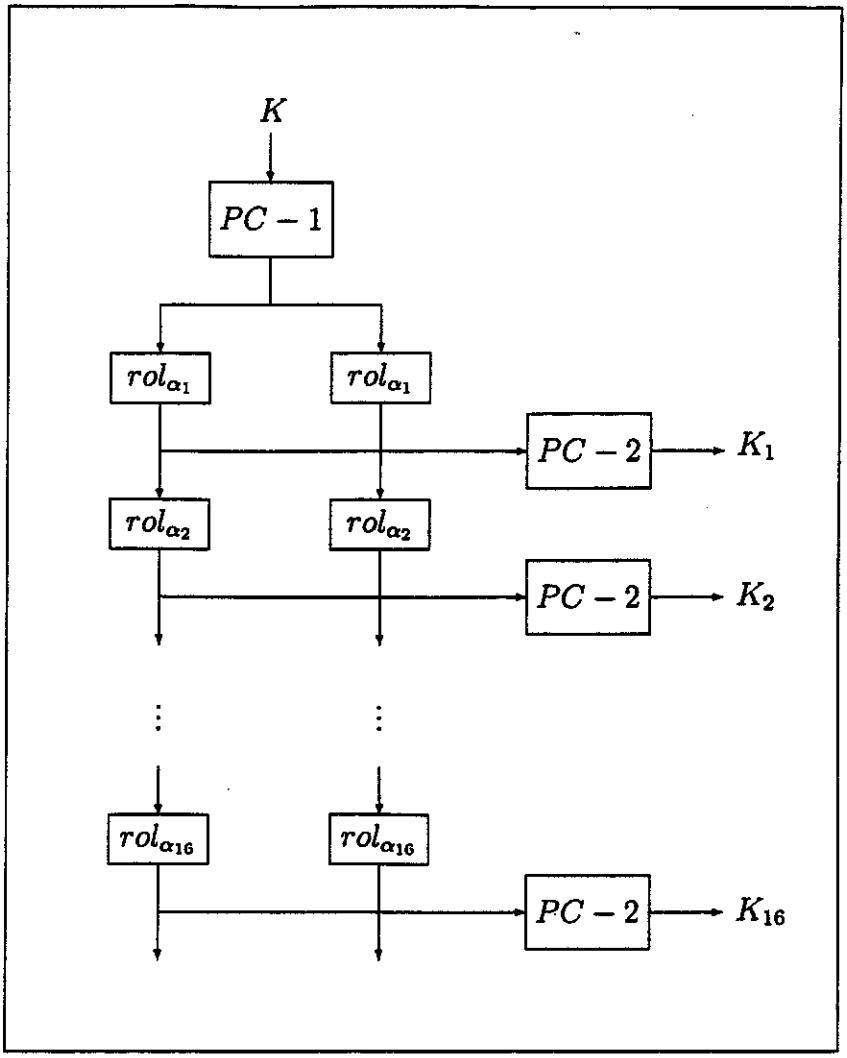


FIG. 2.9 – *Algorithme de cadencement de clé du DES*

Conformément à la description classique du DES, la clé-maître de 56 bits est représentée par 8 mots de 8 octets, le huitième bit de chacun de ces octets étant un bit de parité qui n'est pas utilisé par l'algorithme. Autrement dit, les 56 bits de clé sont numérotés de 1 à 64, sachant que les bits 8, 16, ..., 64 ne sont pas pris en compte.

La permutation $PC-1$ change l'ordre des bits de clés.

$$PC-1 : (x_1 \cdots x_{64}) \longmapsto (y_1 \cdots y_{56}) = (x_{57} x_{49} \cdots x_{12} x_4) \ .$$

Les bits de y correspondent aux bits de x pris dans l'ordre suivant :

$PC-1$						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	44

Après avoir appliqué la permutation $PC-1$, on divise les 56 bits du résultat en deux mots de 28 bits, la partie gauche et la partie droite. Sur chacune de ces deux moitiés, on opère un décalage circulaire vers la gauche de α positions, noté rol_α où la valeur de α dépend du numéro du tour :

tour	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
α_i	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Puis on concaténe les deux mots de 28 bits obtenus après décalage et on applique au résultat la fonction de choix permutée $PC-2$ qui transforme un mot de 56 bits en un mot de 48 bits ; l'ordre des bits de la sortie est donné par :

$PC-2$							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

On vient de donner une description du système de chiffrement que l'on va, entre autre étudier. On développera des algorithmes qui seront systématiquement appliqués sur le DES. Ce système est connu pour être très résistant, il est donc intéressant de faire une analyse cryptographique de ce système. On va maintenant donner une description du sytème de Knudsen et Nyberg. Dans ce cas le shéma de construction est similaire à celui du DES, c'est un Feistel, cependant la fonction itérée est différente.

Le système de chiffrement de Knudsen et Nyberg [14]

Soit la fonction $S : \mathbb{F}_{2^{33}} \rightarrow \mathbb{F}_{2^{33}}$, telle que $S(x) = x^3$. La clef K_i du i-ème tour est un élément de $\mathbb{F}_{2^{33}}$. Soit la fonction $d : \mathbb{F}_{2^{33}} \rightarrow \mathbb{F}_{2^{32}}$ qui extrait un élément de $\mathbb{F}_{2^{32}}$ d'un élément de $\mathbb{F}_{2^{33}}$ par suppression d'un bit. La fonction $e : \mathbb{F}_{2^{32}} \rightarrow \mathbb{F}_{2^{33}}$ étend un mot de 32 bits en un mot de 33 bits en concaténant un bit qui est une combinaison linéaire des bits d'entrée de e. La fonction de tour pour ce système est donc

$$f(x) = d(S(e(x) + K_i))$$

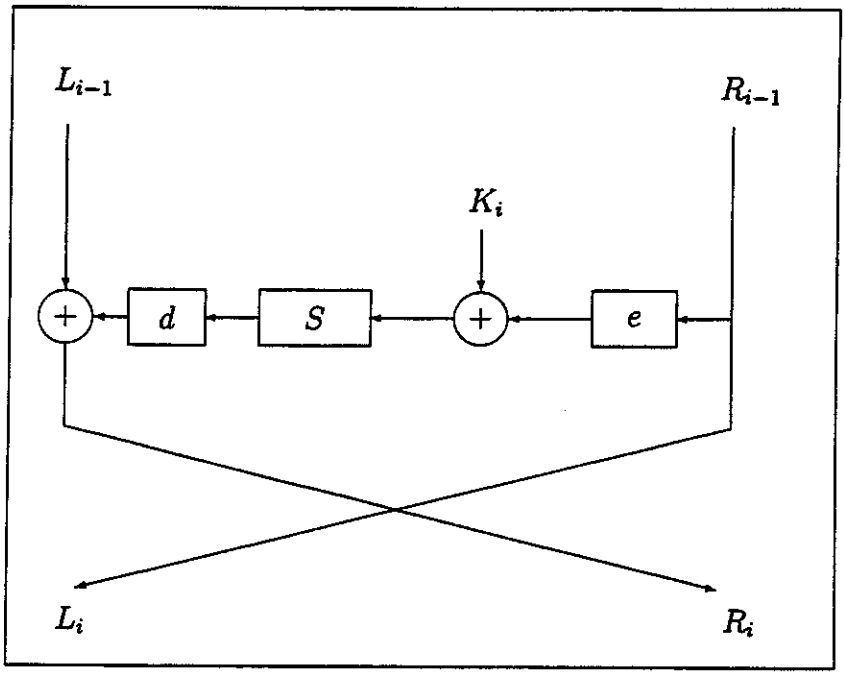


FIG. 2.10 – *Fonction itérée du système de Knudsen et Nyberg*

La fonction itérée F_{K_i} est définie par :

$$\begin{array}{rccc} F_{K_i} : & \mathbf{F}_2^{32} \times \mathbf{F}_2^{32} & \rightarrow & \mathbf{F}_2^{32} \times \mathbf{F}_2^{32} \\ & (L_{i-1}, R_{i-1}) & \mapsto & (L_i, R_i) \end{array}$$

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

Ce système comporte 6 tours de chiffrement par la fonction itéré F_{K_i}. Le cadencement de clé de ce système n'est pas spécifié. Par ailleurs on sait que ce système résiste à la cryptanalyse linéaire et différentielle [14].

Chapitre 3

Quelques attaques connues

Nous allons présenter dans ce chapitre deux types d'attaque déjà connues. Nous allons reprendre brièvement les principes de la cryptanalyse linéaire de Matsui [35]. Cette cryptanalyse est intéressante car elle nous amènera à considérer un problème de décodage, et plus précisément un problème de reconstruction multivarié. Nous allons ensuite décrire la cryptanalyse de Jakobsen. Cette cryptanalyse a permis de mettre en évidence le manque de sécurité du système de Knudsen et Nyberg. Cette attaque repose sur le problème de reconstruction univariée. Nous avons omis d'autres attaques connues telles que la cryptanalyse différentielle. On s'est surtout intéressé aux cryptanalyses qui sont très proches du problème de reconstruction.

3.1 Quelques rappels de cryptographie

Définition 18. *Soit $E = \{(x_i,y_i),i \in \{1,\ldots,k\}\}$ un ensemble de couples clai-rs-chiffrés pour un système itératif de chiffrement par blocs donné. Un algorithme qui distingue avec succès un ensemble de couples clairs-chiffrés d'un ensemble de couples aléatoires est appelé distingueur pour le système de chiffrement considéré.*

Définition 19. *Soit C le chiffré résultant de r tours d'un système de chiffrement itératif par blocs. Soit $\tilde{C}$ le chiffré correspondant résultant des $r-1$ tours du chiffré C. Soit maintenant un ensemble $E = \{(x_i,y_i),i \in \{1,\ldots,k\}\}$ de couples clairs-chiffrés pour le système considéré et un distingueur pour le chiffré réduit $\tilde{C}$. Soit E_{K_r} l'ensemble construit à partir de E en déchiffrant le texte chiffré C résultant d'un tour supplémentaire :*

$$\boxed{\tilde{C}} \xrightarrow{F_{K_r}} \boxed{C} \xrightarrow{F_K} \boxed{Y}$$

soit K_c la clef du dernier tour, et K_w une mauvaise clef. Le distingueur est dit conforme s'il distingue avec succès S_{K_c} de S_{K_w}.

Proposition 5. *Soit C le chiffré résultant de r tours d'un système de chiffrement itératif par blocs. Soit un distingueur conforme pour le chiffré réduit $\tilde{C}$.*

Supposons que ce distingueur requiert k couples de clair-chiffrés et qu'il requiert $t(k)$ opérations. Alors il est possible d'obtenir la clef du dernier tour en $\frac{1}{2}t(k)|\mathcal{K}|$ étapes, ou $\mathcal{K}$ désigne l'espace des clefs du dernier tour.

3.2 Cryptanalyse linéaire

3.2.1 Principe de la cryptanalyse linéaire

La cryptanalyse linéaire introduite par Matsui en 1993 [34] est une attaque à clairs connus qui s'applique aux chiffrements itératifs par blocs. Rappelons que pour un système de chiffrement itératif par bloc, la clef K de t bits (t varie selon le système) induits des clefs de tour K_i comportant t' bits (t' varie selon le système). Supposons que le système chiffre des blocs de u bits. Cette cryptanalyse consiste à approximer la fonction $F : (X,K_i) \mapsto F_{K_i}(X)$ du i-ième tour par une fonction linéaire, c'est-à-dire à trouver $\alpha_i,\beta_i \in \mathbb{F}_2^u$ et $\gamma_i \in \mathbb{F}_2^t$ tels que

$$\beta_i \cdot F_{K_i}(X) = \alpha_i \cdot X + \gamma_i \cdot K \quad \text{avec une bonne probabilité}$$

où $x \cdot y$ est le produit scalaire usuel entre deux vecteurs x et y de $\mathbf{F}_2^t$ ou $\mathbf{F}_2^u$ selon que l'on considère la clef ou le clair. Si l'on a des relations du type $\beta_i = \alpha_{i+1}$, alors on peut chaîner de telles équations en les additionnant simplement et on peut aboutir à une approximation linéaire sur $(r-1)$ tours de l'algorithme de la forme

$$\alpha \cdot Y(0) + \beta \cdot Y(r-1) + \gamma \cdot K = 0,$$

satisfaite avec une bonne probabilité qui permet d'obtenir des informations sur la clé secrète utilisée. On reviendra dans la section 6 du chapitre 6 sur le calcul de la probabilité d'exactitude de cette relation.

Pour retrouver la sous-clé du dernier tour, on utilise une expression linéaire approximant les $(r-1)$ premiers tours du chiffrement itératif, c'est à dire telle que celle que l'on vient d'obtenir :

$$\alpha \cdot Y(0) + \beta \cdot Y(r-1) = \gamma \cdot K$$

satisfaite avec une probabilité p où $|p - \frac{1}{2}|$ est élevée. Comme la fonction itérée F est une permutation, la sortie $Y(r-1)$ de l'avant-dernier tour est uniquement déterminée par le chiffré $Y(r)$ et la clé K_r. Cette expression s'écrit alors :

$$\alpha \cdot Y(0) + \beta \cdot F^{-1}(Y(r),K_r) = \gamma \cdot K.$$

Pour un certain nombre N de couples clairs - chiffrés $(Y(0),Y(r))$, on va donc calculer

$$\alpha \cdot Y(0) + \beta \cdot F^{-1}(Y(r),k_r)$$

pour toutes les valeurs k_r possibles de la clé K_r. Si la valeur testée k_r est exacte, alors le nombre de couples $(Y(0),Y(r))$ tels que

$$\alpha \cdot Y(0) + \beta \cdot F^{-1}(Y(r),k_r) = 0 \tag{3.1}$$

est proche de Np si $\gamma \cdot K = 0$ ou de $N(1-p)$ si $\gamma \cdot K = 1$. Il est donc relativement éloigné de $\frac{N}{2}$ dès lors que $|p - \frac{1}{2}|$ est élevée. Par contre si la valeur k_r testée ne correspond pas à celle qui est employée, $F^{-1}(Y(r),k_r)$ est aléatoire (hypothèse de wrong key randomization [11]). Le nombre de couples $(Y(0),Y(r))$ vérifiant (3.1) est donc proche de $\frac{N}{2}$.

Grâce à ces propriétés on peut donc, en essayant successivement toutes les valeurs possibles pour la clé du dernier tour, déterminer celle qui a été employée dans le système (et un bit d'information supplémentaire donné par la valeur de $\gamma \cdot K$).

Cette attaque se déroule donc de la manière suivante :

1. Pour chacun des N couples $(Y(0),Y(r))$
   Pour chaque valeur k_r de la clé au dernier tour
   Si $\alpha \cdot Y(0) + \beta \cdot F^{-1}(Y(r),k_r) = 0$
   incrémenter le compteur $c[k_r]$.
2. Les valeurs les plus probables pour K_r sont celles qui maximisent $|c[k_r] - \frac{N}{2}|$.
3. En fonction du signe de $(c[k_r] - \frac{N}{2})$ et de $(p - \frac{1}{2})$, en déduire la valeur de $\gamma \cdot K$.

Le coût de fonctionnement de cet algorithme dépend alors du nombre de couples clair-chiffré considérés. Il s'agit de savoir quel est la taille de l'échantillon nécessaire pour approximer un évènement qui se produit avec une probabilité $p = 1/2 + \epsilon$ avec une erreur d'au plus e. Ce sont des résultats d'échantillonage qui sont rappelés dans le chapitre 4.

Proposition 1. [34]
Soit N le nombre de couples clair-chiffré connus (où les textes clairs sont aléatoires et uniformément distribués).
Soit p la probabilité que

$$\alpha \cdot Y(0) + \beta \cdot Y(r-1) = \gamma \cdot K.$$

Alors le taux de succès de l'algorithme est

$$\int_{-2\sqrt{N}|p-\frac{1}{2}|}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-\frac{x^2}{2}) dx.$$

Le tableau de la figure 3.1 donne une estimation de ce taux de succès pour quelques valeurs de N.

N	$\frac{1}{4}\lvert p-\frac{1}{2}\rvert^{-2}$	$\frac{1}{2}\lvert p-\frac{1}{2}\rvert^{-2}$	$\lvert p-\frac{1}{2}\rvert^{-2}$	$2\lvert p-\frac{1}{2}\rvert^{-2}$
taux de succès	84,1 %	92,1 %	97,7 %	99,8 %

FIG. 3.1 - *Taux de succès*

De façon identique, on peut utiliser une approximation linéaire des $(r-2)$ tours centraux du chiffrement :

$$\alpha \cdot Y(1) + \beta \cdot Y(r-1) = \gamma \cdot K.$$

et remplacer $Y(1)$ par $F(Y(0),K_1)$ et $Y(r-1)$ par $F^{-1}(Y(r),K_r)$. On obtient donc une approximation de la forme

$$\alpha \cdot F(Y(0),K_1) + \beta \cdot F^{-1}(Y(r),K_r) = \gamma \cdot K \tag{3.2}$$

Comme dans l'algorithme précédent, on essaie successivement toutes les valeurs possibles pour (K_1,K_r) et on parvient ainsi à déterminer les valeurs de ces deux clés qui sont effectivement utilisées. De manière générale, cette méthode n'est pas efficace puisqu'elle revient à tester toutes les clefs possibles pour (K_1,K_r), cependant il est possible, suivant le système considéré, et suivant les approximations linéaires dont on dispose, que peu de bits de clef soient impliqués dans l'equation 3.2, donc cette méthode dépend largement du système utilisé et des équations linéaires connues que l'on a.

3.2.2 Recherche d'expressions linéaires pour le DES

Pour trouver des expressions linéaires du DES, il faut par exemple rechercher des approximations linéaires des boîtes S qui sont satisfaites avec une probabilité la plus éloignée possible de 0,5.

La meilleure approximation linéaire d'une boîte S concerne la boîte S_5 [34]. En effet, si x_1,x_2,x_3,x_4,x_5,x_6 sont les entrées de S_5 et y_1,y_2,y_3,y_4 ses sorties, on a

$$x_2 = y_1 + y_2 + y_3 + y_4$$

avec une probabilité $\frac{3}{16}$. En notant X, le mot de $\mathbf{F}_2^{48}$ en entrée de S, on a donc avec probabilité $\frac{3}{16}$:

$$S(X)[17] + S(X)[18] + S(X)[19] + S(X)[20] = X[26].$$

La permutation P transforme les bits 17, 18, 19 et 20 en 3, 8, 14 et 25, et le bit 26 d'un mot étendu par E correspond à son bit 17. On en déduit donc, pour $Z_i = P(S(E(R_{i-1}) + K_i))$,

$$Z_i[3] + Z_i[8] + Z_i[14] + Z_i[25] = K_i[26] + R_{i-1}[17].$$

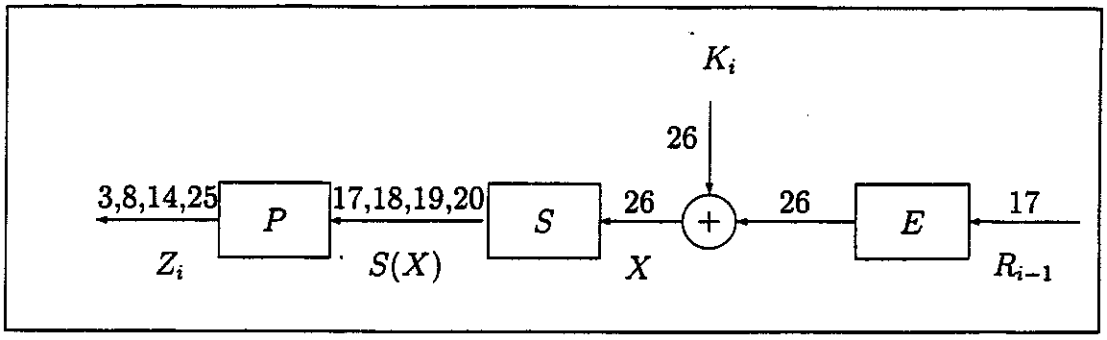


FIG. 3.2 – *Approximation linéaire d'un tour du DES*

Comme $Z_i = L_{i-1} + R_i$, on en déduit l'approximation linéaire suivante du i-ième du DES :

$$L_{i-1}[3,8,14,25] + R_{i-1}[17] + R_i[3,8,14,25] = K_i[26]$$

satisfaite avec la probabilité $\frac{3}{16}$. On note dorénavant $X[i_1, \cdots, i_n] = \sum_{j=1}^{n} X[i_j]$.

Appliquons maintenant cette approximation linéaire aux premier et troisième tours du DES. On a pour le premier tour

$$L_0[3,8,14,25] + R_0[17] + R_1[3,8,14,25] = K_1[26]$$

et pour le troisième tour

$$L_2[3,8,14,25] + R_2[17] + R_3[3,8,14,25] = K_3[26].$$

Comme $R_2 = L_3$ et $L_2 = R_1$, on en déduit l'approximation linéaire suivante des 3 premiers tours du DES :

$$L_0[3,8,14,25] + R_0[17] + L_3[17] + R_3[3,8,14,25] = K_1[26] + K_3[26].$$

Cette équation est satisfaite avec la probabilité

$$p = \left(\frac{12}{64}\right)^2 + \left(1 - \frac{12}{64}\right)^2 = 0{,}7.$$

Cette approximation linéaire du DES à 3 tours permet donc de cryptanalyser le DES à 4 tours : on peut en effet déterminer K_4 et $K_1[26] + K_3[26]$ (*i.e.* un bit d'information de la clé secrète K utilisée) avec un taux de succès de 97,7 % à partir de N couples clair-chiffré connus (3.1) où

$$N = \left|p - \frac{1}{2}\right|^{-2} = 25.$$

Pour cryptanalyser le DES complet de cette façon, on est donc amené à rechercher une "bonne" approximation linéaire des 15 premiers tours du DES. Sur 15 tours, la meilleure approximation linéaire est [34]:

$$L_0[8,14,25] + R_0[16,20] + L_{15}[3,8,14,25] + R_{15}[17] = K_1[25,27] + K_3[26] + K_4[4] + K_5[26] + K_7[26] + K_8[4] + K_9[26] + K_{11}[26] + K_{12}[4] + K_{13}[26] + K_{15}[26].$$

Elle est satisfaite avec une probabilité

$$p = \frac{1}{2} - 1{,}19 \cdot 2^{-22}.$$

En utilisant cette approximation linéaire, on peut donc déterminer la clé du dernier tour ainsi qu'un bit d'information de la clé (donné par le membre de droite de l'approximation) avec un taux de succès de 97,7 % pour 2^{43} couples clair-chiffré connus.

La table 3.1 donne la probabilité p de la meilleure approximation linéaire de r tours du DES pour $4 \leq r \leq 18$, et le nombre N de couples clair-chiffré nécessaires pour retrouver un bit d'information de la clé avec un taux de succès de 97,7 %.

nb de tours	4	5	6	7	8
$p - \frac{1}{2}$	$-1.95 \cdot 2^{-5}$	$1.22 \cdot 2^{-6}$	$-1.95 \cdot 2^{-9}$	$1.95 \cdot 2^{-10}$	$-1.22 \cdot 2^{-11}$
N	2^8	2^{11}	2^{16}	2^{18}	2^{21}

nb de tours	9	10	11	12
$p - \frac{1}{2}$	$-1.91 \cdot 2^{-14}$	$-1.53 \cdot 2^{-15}$	$1.91 \cdot 2^{-16}$	$-1.19 \cdot 2^{-17}$
N	2^{26}	2^{29}	2^{30}	2^{33}

nb de tours	13	14	15	16
$p - \frac{1}{2}$	$-1.49 \cdot 2^{-19}$	$-1.19 \cdot 2^{-21}$	$1.19 \cdot 2^{-22}$	$-1.49 \cdot 2^{-24}$
N	2^{37}	2^{41}	2^{43}	2^{47}

nb de tours	17	18
$p - \frac{1}{2}$	$-1.16 \cdot 2^{-26}$	$-1.86 \cdot 2^{-28}$
N	2^{51}	2^{54}

TAB. 3.1 – *Probabilités des meilleurs approximations linéaires du DES en fonction du nombre de tours*

L'attaque linéaire la plus efficace sur le DES est due à Matsui [35]. Elle utilise une approximation linéaire sur les 14 tours centraux du DES afin de déterminer les clés utilisées aux premier et dernier tours. L'approximation linéaire utilisée

est relativement creuse et surtout implique peu de boîtes S. Ainsi pour cette approximation linéaire, seules quelques bits des clefs du premier et dernier tour sont impliquées. On trouve ainsi, à partir de 2^{41} couples clairs-chiffrés, 26 des 56 bits de la clé secrète. Les 30 derniers bits sont déterminés par recherche exhaustive. Il s'agit de la meilleure attaque actuellement connue sur le DES. Elle requiert de l'ordre de 2^{43} chiffrements DES. Pour des résultats plus récents concernant la réelle complexité de ces attaques, nous renvoyons à [31].

3.2.3 Les liens avec les problèmes de décodage

On a vu dans la section précédente que la meilleure attaque sur le DES à 16 tours repose sur la recherche d'expressions linéaires du type

$$\alpha \cdot Y(0) + \beta \cdot Y(r-1) + \gamma \cdot K = 0$$

qui sont vérifiées avec une bonne probabilité. On supposera que la clef K comporte t bits et que le système utilisé chiffre des blocs de u bits. Par construction la variable $Y(r-1)$ dépend des variables $Y(0)$ et K. Dans la suite nous considèrerons des algorithmes de reconstruction multivariée qui nécessitent la connaissance de certaines entrées choisies de la fonction booléenne considérée. Ainsi, nous fixerons β pour nous ramener à un problème plus simple de décodage. Soit la fonction booléenne $f : \mathbb{F}_2^{u+t} \rightarrow \mathbb{F}_2$ telle que

$$f(Y(0),K) = \beta \cdot Y(r-1),$$

alors trouver la fonction linéaire qui coïncide avec f sur la plus grande fraction des entrées de $\mathbb{F}_2^{u+t}$ donnera la meilleure approximation linéaire à β choisi du type

$$\alpha \cdot Y(0) + \beta \cdot Y(r-1) + \gamma \cdot K = 0.$$

Nous développerons dans le chapitre "Tests et reconstruction multivariés" des algorithmes permettant d'effectuer ce type de décodage dans $R[1,m]$.

3.3 Cryptanalyse de Jakobsen

On va résumer dans cette section les travaux de Thomas Jakobsen [30]. Considérons r tours d'un système itératif de chiffrement par blocs dont la fonction itérée sera notée F_{K_i} et vérifiera

$$C_i = F_{K_i}(C_{i-1}),$$

ou $C_0 = (L_0,R_0)$ est le texte clair, K_i la clef du i-ème tour, et $C_r = (L_r,R_r)$ le texte chiffré. On supposera que F_{K_i} est une bijection prenant ses entrées dans $\mathbb{F}_{2^n}$. On supposera de plus que les clefs de chaque tour sont indépendantes et uniformément distribuées. Thomas Jakobsen dit dans [30] que considérer que

le texte clair et les clefs sont des variables aléatoires indépendantes implique que les entrées de chaque tour peuvent être considérées comme des variables aléatoires indépendantes. Nous verrons lors de l'analyse cryptographique du DES (cf chapitre 6, section 6) que cette considération est très exagérée.

Proposition 6. *[30]. Reprenons les notations précédentes et considérons une fonction itérée F_{K_i}. Supposons qu'il existe un polynôme $P : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ de degré m tel que*

$$\underset{x \in \mathbb{F}_{2^n}}{\mathbf{Prob}} [P(x) = F_{K_i}(x)] = \mu,$$

alors la fonction de chiffrement sur r tours $F = F_{K_r} \circ F_{K_{r-1}} \circ \ldots\ldots \circ F_{K_1}$ est telle qu'il existe un polynôme $Q : \mathbb{F}_{2^n} \rightarrow \mathbb{F}_{2^n}$ de degré au plus m^r qui vérifie

$$\underset{x \in \mathbb{F}_{2^n}}{\mathbf{Prob}} [Q(x) = F(x)] \geq \mu^r$$

Remarque 8. *En pratique le distingueur seul ne permet pas d'attaquer les systèmes de chiffrement itératifs par blocs car l'espace des clefs est trop grand, cependant l'existence d'un polynôme qui approxime le système est en général suffisante pour considérer ce système comme cassé puisqu'elle permet d'inverser le système et de retrouver dans une certaine proportion le texte clair.*

Théorème 7. *[30]. Soit C une fonction de chiffrement telle qu'il existe au moins un polynôme P de degré au plus m tel que*

$$\underset{x \in \mathbb{F}_{2^n}}{\mathbf{Prob}} [P(x) = C(x)] = \mu,$$

alors le système peut être cassé en utilisant

$$n = \frac{m}{\mu^2}$$

couples de clair-chiffré connus avec une complexité polynomiale en n.

Preuve : C'est le théorème de Sudan [45] (chapitre 1, théorème 2) appliqué à un système de chiffrement. □

Rappelons brièvement comment est construite la fonction de chiffrement du système de Knudsen et Nyberg. Soit la fonction $S : \mathbb{F}_{2^{33}} \rightarrow \mathbb{F}_{2^{33}}$, telle que $S(x) = x^3$. La clef K_i du i-ème tour est un élément de $\mathbb{F}_{2^{33}}$. Soit la fonction $d : \mathbb{F}_{2^{33}} \rightarrow \mathbb{F}_{2^{32}}$ qui extrait un élément de $\mathbb{F}_{2^{32}}$ d'un élément de $\mathbb{F}_{2^{33}}$ par suppression d'un bit. La fonction $e : \mathbb{F}_{2^{32}} \rightarrow \mathbb{F}_{2^{33}}$ étend un mot de 32 bits en un mot de 33 bits en concaténant un bit qui est une combinaison linéaire des bits d'entrée de e. La fonction de tour pour ce système est donc

$$f(x) = d(S(e(x) + K_i)),$$

les équations suivantes donnent donc le chiffré :

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

On définit alors une variante de ce système qui prend en entrée les éléments $(L_0,R_0) \in \mathbb{F}_{2^{33}}^2$ et qui retourne des éléments $(L_r,R_r) \in \mathbb{F}_{2^{33}}^2$ pour r tours :

$$\begin{aligned} D_0^L &= d(L_0) \\ D_0^R &= d(R_0) \\ D_i^L &= D_{i-1}^R \\ D_i^R &= F_{K_i}(D_{i-1}^R) + D_{i-1}^L \\ L_r &= e(D_r^L) \\ R_r &= e(D_r^R), \end{aligned}$$

en d'autres termes cette variante est équivalente au système de Knudsen et Nyberg qui à été composé avec les fonctions d et e. Clairement, si l'on peut casser ce système, alors on peut casser le système de Knudsen et Nyberg. Par construction, la fonction $e(d(S(x)))$ vérifie :

$$\underset{x \in \mathbb{F}_{2^{64}}}{\mathbf{Prob}} \left[e(d(S(x))) = x^3 \right] = \frac{1}{2}.$$

En conséquence, on peut utiliser l'algorithme de Guruswami-Sudan [17] (chapitre 1, théorème 2) pour attaquer ce système. Pour 6 tours de chiffrement d'après la proposition 5, il existe un polynôme p de degré au plus $3^{6-2} = 81$ qui coïncide avec la fonction itérée sur une fraction d'au moins $\mu = 2^{-(6-2)} = 1/16$ des entrées. Pour trouver l'approximation polynomiale univariée du système de Knudsen et Nyberg à 6 tours il suffit de

$$n = \frac{2 \times 81}{(1/16)^2} < 2^{16}$$

couples de clairs-chiffrés connus. Cette attaque permet d'affaiblir considérablement le système puisqu'elle permet de retrouver le clair avec une probabilité de succès de $1/16$.

La conclusion

On a vu que la simple connaissance d'un polynôme de bas degré approximant le système de chiffrement suffisait pour considérer ce système comme cassé. Donc il serait intéressant de développer des algorithmes moins coûteux que l'algorithme de Sudan-Guruswami [17] qui testeraient simplement l'existence de tels polynômes.

Chapitre 4

Quelques rappels de probabilités

Nous allons juste ici faire les rappels de probabilités nécessaire à la bonne compréhension des prochains chapitres. Le but est d'exposer les principaux résultats connus concernant l'échantillonnage. Tous ces résultats peuvent être trouvés dans les ouvrages classiques, comme par exemple [29].

4.1 Quelques lois de probabilité

On va considérer le cas le plus simple. Supposons que notre espace probabilisable Ω soit l'union disjointe de deux événements : un événement et son contraire, respectivement A et A^c. On a naturellement

$$\mathbf{Prob}\,[A] + \mathbf{Prob}\,[A^c] = 1.$$

On note $\mathbf{Prob}\,[A] = \lambda$ et $\mathbf{Prob}\,[A^c] = \overline{\lambda} = 1 - \lambda$.

Définition 20. *Reprenons les notations ci-dessus. On appelle variable de Bernoulli, l'application $X : \Omega \to \mathbb{N}$ qui vérifie*

$$X(a) = \begin{cases} 1 & \text{si } a \in A \\ 0 & \text{si } a \in A^c \end{cases}$$

Définition 21. *On considère une urne contenant b boules blanches et $T - b$ boules noires. On en tire au hasard N avec remise. Soit X la variable aléatoire représentant le nombre de boules blanches tirées. On appelle loi binomiale la loi que suit la variable X. En posant $\lambda = \frac{b}{T}$, on voit que :*

$$\mathbf{Prob}\,[X = m] = \binom{N}{m} \cdot \lambda^m \cdot (1 - \lambda)^{N-m},$$

cette loi se note $B(\lambda,N)$.

Définition 22. *On dit que deux variables aléatoires X et Y sont indépendantes si*

$$\mathbf{Prob}[Y \mid X] = \mathbf{Prob}[Y],$$

la probabilité conditionnelle $\mathbf{Prob}[Y \mid X]$ *est définie par*

$$\frac{\mathbf{Prob}[X,Y]}{\mathbf{Prob}[X]}.$$

Proposition 7. *Soit Ω un espace probabilisable. Soient $X_1,\ldots,X_N$ N variables aléatoires de Bernouilli indépendantes telles que :*

$$\mathbf{Prob}\,[X_i = 1] = \lambda,\ i \in [1...N],$$

$$\mathbf{Prob}\,[X_i = 0] = 1 - \lambda,\ i \in [1...N],$$

alors la variable aléatoire

$$S_N = X_1 + ... + X_N,\ X \in [0,...,N]$$

suit la loi binomiale $B(\lambda,N)$.

4.2 Quelques quantités à connaître

Définition 23. *Soit Ω un espace probabilisable. Soit X une variable aléatoire prenant pour valeurs $x_1,x_2,...$ avec les probabilités respectives $p(x_1),p(x_2),....$ L'espérance de X est définie par la quantité :*

$$\mathbf{E}(X) = \sum_k x_k p(x_k).$$

Si cette somme converge absolument, on dit que X possède une espérance finie. Si $\sum |x_k|p(x_k)$ diverge, on dit que X n'a pas d'espérance finie. Habituellement on note $\mu = \mathbf{E}(X)$.

Théorème 8. *Toute fonction $\phi(x)$ définit une nouvelle variable aléatoire $\phi(X)$. Si $\phi(X)$ admet une espérance finie, alors*

$$\mathbf{E}(\phi(X)) = \sum_k \phi(x_k)p(x_k);$$

la série converge absolument si et seulement si $\mathbf{E}(\phi(X))$ existe. En particulier, pour toute constante a, on a : $\mathbf{E}(aX) = a\mathbf{E}(X)$.

Théorème 9. *Si $X_1,...,X_N$ sont des variables aléatoires à espérance finie, alors l'espérance de leur somme existe et c'est précisément la somme de leur espérance :*

$$\mathbf{E}(X_1 + ... + X_N) = \mathbf{E}(X_1) + ... + \mathbf{E}(X_N).$$

Théorème 10. *Soient X et Y deux variables aléatoires indépendantes à espérance finie. Alors leur produit est une variable aléatoire à espérance finie et*

$$\mathbf{E}(XY) = \mathbf{E}(X)\mathbf{E}(Y).$$

Exemple 5 :

Soient $X_1,...,X_N$ des variables aléatoires de Bernoulli indépendantes, telles que

$$\mathbf{Prob}[X_i = 1] = \lambda,\ i \in [1...N],$$

alors $\mathbf{E}(X_i) = 0 \cdot (\lambda - 1) + 1 \cdot \lambda = \lambda$. D'après la proposition 7, la variable aléatoire

$$S_N = X_1 + ... + X_N$$

suit la loi binomiale et d'après le théorème 7, la valeur de son espérance est

$$\mathbf{E}(S_N) = \mathbf{E}(X_1) + ... + \mathbf{E}(X_N) = N\lambda.$$

◊

Définition 24. *Soit X une variable aléatoire à espérance finie et telle que X^2 soit également à espérance finie, alors on appelle variance la quantité*

$$\mathbf{Var}(X) = \mathbf{E}((X - \mathbf{E}(X))^2) = \mathbf{E}((X - \mu)^2) = \mathbf{E}(X^2) - \mu^2.$$

La racine carrée de cette quantité est appelée l'écart-type.

Théorème 11. *Soient $X_1,...,X_N$ des variables aléatoires indépendantes admettant une variance. Soit la variable aléatoire $S_N = X_1 + ... + X_N$. Alors on a*

$$\mathbf{Var}(S_N) = \mathbf{Var}(X_1) + ... + \mathbf{Var}(X_N).$$

Cette formule ne tient pas toujours si les variables sont dépendantes.

Exemple 5 :

Soient $X_1,...,X_N$ des variables aléatoires de Bernoulli indépendantes, telles que

$$\mathbf{Prob}[X_i = 1] = \lambda,\ i \in [1...N],$$

alors

$$\mathbf{Var}(X_i) = 0^2 \cdot (1 - \lambda) + 1^2 \cdot \lambda - \lambda^2 = \lambda\overline{\lambda}.$$

On en déduit donc la valeur de la variance de $S_N = X_1 + ... + X_N$:

$$\mathbf{Var}(S_N) = N\lambda\overline{\lambda}.$$

◊

4.3 L'échantillonnage

Théorème 12. *(Inégalité de Chebyshev) Soient Ω un espace probabilisable, X une variable aléatoire à espérance et à variance finies. Pour tout $s > 0$*

$$\mathbf{Prob}[|X| \geq s] \leq s^{-2}\mathbf{E}(X^2).$$

En particulier, si $\mathbf{E}(X) = \mu$ alors

$$\mathbf{Prob}[|X - \mu| \geq s] \leq s^{-2}\mathbf{Var}(X),$$

où $||$ désigne la valeur absolue.

Preuve : La seconde inégalité est obtenue en appliquant la première à la variable $X - \mu$. En utilisant les notations précédentes on a

$$\mathbf{Prob}[|X| \geq s] = \sum_{|x_i| \geq s} p(x_i) \leq s^{-2} \sum_{|x_i| \geq s} x_i^2 p(x_i) \leq s^{-2}\mathbf{E}(X^2).$$

□

Remarque 9. *L'inégalité de Chebyshev donne une majoration de la probabilité de succès de l'évènement $|X| \geq s$ ou bien $|X - \mu| \geq s$ suivant le cas. De manière équivalente cette inégalité donne une minoration de la probabilité d'échec, tous simplement par-ce-que* $\mathbf{Prob}[A] = 1 - \mathbf{Prob}[A^c]$.

Exemple 6 :

Soient $X_1,...,X_N$ des variables aléatoires de Bernoulli indépendantes, telles que

$$\mathbf{Prob}[X_i = 1] = \lambda,\ i \in [1...N],$$

Soit la variable aléatoire $S_N = X_1 + ... + X_N$. Dans les exemples précédents, on a déjà déterminé les valeurs de l'espérance et de la variance de S_N, on peut donc appliquer l'inégalité de Chebyshev à cette variable :

$$\mathbf{Prob}[|X - N\lambda| \geq s] \leq s^{-2}N\lambda(1 - \lambda).$$

Maintenant posons $s = N\beta$, on obtient alors une expression qui sera très utile par la suite :

$$\mathbf{Prob}\left[\left|\frac{X_1 + ... + X_N}{N} - \lambda\right| \geq \beta\right] \leq \frac{\lambda(1 - \lambda)}{N\beta^2}$$

◊

L'inégalité de Chebyshev nous donne une majoration qui reste cependant assez imprécise. Nous allons donc donner des résultats plus forts que l'inégalité de Chebyshev.

Définition 25. *La fonction définie par*

$$\varkappa(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

est appelée densité normale; son intégrale

$$\mathcal{N}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}y^2} dy$$

est appelée distribution normale.

La fonction $\varkappa(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$ est bien entendu une densité de probabilité. On peut aisément vérifier que

$$\int_{-\infty}^{+\infty} \varkappa(x)dx = 1,$$

ce qui implique

$$\mathcal{N}(-x) = 1 - \mathcal{N}(x).$$

Lemme 1. *Reprenons les notations précédentes. Alors on a :*

$$(x^{-1} - x^{-3})\varkappa(x) < 1 - \mathcal{N}(x) < x^{-1}\varkappa(x), \quad x > 0.$$

Preuve : On montre aisément que

$$(1 - 3x^{-4})\varkappa(x) < \varkappa(x) < (1 + x^{-2})\varkappa(x).$$

par intégration entre 0 et l'infini, on obtient le résultat. □

Théorème 13. *(Théorème central limite) Soient X_1,...,X_N des variables aléatoires indépendantes ayant même distribution. Supposons que chacune de ces variables aléatoires admette une espérance finie $\mu = \mathbf{E}(X_k)$ et une variance finie $\sigma^2 = \mathbf{Var}(X_k)$. Soit alors la variable aléatoire $S_N = X_1 + ... + X_N$. Alors pour tout β fixé, on a la limite suivante*

$$\mathbf{Prob}\left[\frac{S_N - N\mu}{\sigma\sqrt{N}} < \beta\right] \xrightarrow[\infty]{N} \mathcal{N}(\beta), \tag{4.1}$$

où $\mathcal{N}$ est la distribution normale introduite précédemment.

Exemple 7 :

Voici quelques résultats très pratiques sur l'inégalité de Hoeffding.

Soient X_1,...,X_N des variables aléatoires indépendantes, telles que

$$\mathbf{E}_{\mathbf{X_i}}(X_i) = \lambda,\ i \in [1...N] \quad \text{et } a \leq X_i \leq b.$$

Soit la variable aléatoire $S_N = X_1 + ... + X_N$.

$$\mathbf{Prob}\left[\left|\frac{S_N}{N} - \lambda\right| > \alpha\right] \leq 2e^{\frac{-2N\alpha^2}{(a-b)^2}} \tag{4.2}$$

◊

Le théorème suivant est la conséquence directe de la formule 4.2 :

Théorème 14. *Soit un événement A qui se produit avec une probabilité λ. Supposons que l'on veuille approcher cette probabilité λ avec une précision α, et avec une probabilité de succès au moins égale à $1 - S$ Alors une taille d'échantillon de l'ordre de*

$$N = \mathcal{O}\left(\alpha^{-2} \log(1/S)\right)$$

conviendra.

Une autre application qui reviendra par la suite est la détection, on se posera alors la question suivante : sachant qu'un événement A se produit avec probabilité λ, quel taille d'échantillon doit-on considérer pour que l'evènement A se produise au moins une fois? Pour répondre à la question il suffit de reprendre la formule 4.2 et de l'appliquer au paramtre $\alpha = \lambda - \epsilon$ avec $\epsilon > 0$. On obtient alors la formule suivante pour N suffisamment grand :

$$\mathbf{Prob}\left[\left|\frac{S_N}{N} - \lambda\right| > \lambda - \epsilon\right] \leq 2e^{-2N(\lambda-\epsilon)^2} \tag{4.3}$$

La conséquence directe de la formule 4.3 se traduira par le théorème suivant :

Théorème 15. *Soit un évènements A qui se produit avec une probabilité λ. Alors une taille d'échantillon de l'ordre de*

$$N = \mathcal{O}\left(\log(1/S)\right)$$

est suffisante pour que l'évènement A se produise au moins une fois et avec une probabilité de succès au moins égale à $1 - S$.

Un cas concret

Soit une urne contenant des boules blanches et des boules noires. On effectue des tirages avec remise. Soit λ, la probabilité de tirer une boule blanche et $1 - \lambda$ la probabilité de tirer une boule noire. On aimerait alors répondre à la question suivante : quel nombre de tirages doit on effectuer pour que la proportion de boules tirées soit égale à λ avec une erreur d'au plus x pour cent de λ? Si la précision demandée est suffisante, c'est-à-dire si cette précision nécessite plus d'une trentaine de tirages, la distribution normale répond directement à la question, d'autre part les formules précédentes indiquent que si la précision désirée α tend vers 0, alors l'inégalité de Chebyshev devient une bonne approximation et la taille de l'échantillon est alors de l'ordre de $\mathcal{O}(1/\alpha^2)$. Les inégalités que l'on a données fournissent une une majoration de la probabilité d'échec.

Chapitre 5

Test et reconstruction univarié

Ce chapitre est fortement inspiré du chapitre 3 de la thèse de M. Sudan [44] sur les tests de bas degré. Ses résultats sont applicables sur les corps premiers $\mathbb{F}_p$. Ses résultats vont être ici généralisés aux extensions du corps premier $\mathbb{F}_2$ [48].

5.1 Les problèmes considérés

On veut étudier des fonctions $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$ ($q = 2^t$). On considère ces fonctions comme des boites noires pour lesquelles on ne connaît qu'une fraction des entrées-sorties. Les entrées que l'on considérera seront de deux types :

- Les entrées de type clairs-connus que l'on a définies dans le chapitre "Rappel sur le codage", c'est-à-dire que l'on suppose qu'une fraction aléatoire d'entrées-sorties nous est donnée.
- Les entrées de type clairs-choisis que l'on a définies dans le chapitre "Rappel sur le codage", c'est-à-dire que l'on choisit le type d'entrée que l'on désire, par exemple des éléments de $\mathbb{F}_q$ dont le premier bit est égal à 1.

On s'intéressera alors à trois types de problèmes :

- Le problème *"Poly-agree"* que nous allons définir. En particulier, un algorithme qui répond à ce problème, prend en entrée les entiers k,T,l et un ensemble de couples pris aléatoirement :

  $$\{(x_1,y_1),\ldots,(x_l,y_l)\}$$

  tels que pour tout $i \in [1,\ldots,l]$ $x_i \in \mathbb{F}_q$, $y_i \in \mathbb{F}_q$. Cet algorithme doit ensuite répondre à la question : existe-t-il au moins un polynôme $P : \mathbb{F}_q \rightarrow \mathbb{F}_q$ de degré au plus k pour lequel $P(x_i) = y_i$ pour au moins T valeurs de i distinctes? Ce sont ces types d'algorithmes que nous appellerons des testeurs. Lorsque $\mathbb{F}_q$ est un corps premier, M. Sudan a développé dans sa thèse [44] de tels testeurs. Nous étendrons le domaine d'application de ces tests aux corps $\mathbb{F}_q$, avec $q = 2^t$. Sudan a proposé dans sa thèse de

tels algorithmes dans deux cadres différents. on étudiera d'abord un type d'algorithmes que l'on peut qualifier d'algorithmes à clairs choisis, puisque l'ensemble de couples

$$\{(x_1,y_1),\ldots,(x_l,y_l)\}$$

est cette fois-ci choisi. On étendra alors les résultats de M. Sudan qui sont applicables sur des corps de type $\mathbb{F}_p$ aux extension de corps du type $\mathbb{F}_q$, avec $q = 2^t$.

- Dans une deuxième partie on étudiera le même problème, mais dans le cadre clairs connus, on proposera alors un algorithme simple pour effectuer ce type de test, qui a pour principale intérêt de montrer le lien qu'il y a entre le problème *"Poly-agree"* et le problème d'estimation d'une distance et le problème de reconstruction, mais on ne montrera pas que ces problèmes sont équivalents.
- Le problème de reconstruction qui est défini dans le chapitre 1, section 7. En particulier, on aimerait construire des algorithmes répondant à ce problème, c'est-à-dire des algorithmes prenant en entrée les entiers k,T,l et un ensemble de couples pris aléatoirement :

$$\{(x_1,y_1),\ldots,(x_l,y_l)\}$$

tel que pour tout $i \in [1,\ldots,l]$ $x_i \in \mathbb{F}_q$, $y_i \in \mathbb{F}_q$. Cet algorithme doit ensuite retourner la liste de tous les polynômes $P : \mathbb{F}_q \rightarrow \mathbb{F}_q$ de degré au plus k pour lesquels $P(x_i) = y_i$ pour au moins T valeurs de i distinctes. On aimerait alors s'inspirer des test que l'on a étudier précédemment On a vu qu'un algorithme très efficace, dû à M. Sudan [45], pouvait répondre à ce problème dans les conditions qu'il définit.

5.1.1 Les motivations

Le but est de pouvoir construire des distingueurs efficaces pour les systèmes de chiffrement par bloc. On a vu dans le chapitre 3, section 3 que si l'on était capable de répondre favorablement à l'un des problèmes que l'on considère, alors on obtiendrait naturellement un distingueur. Jakobsen a montré que le système de Knudsen et Nyberg ne résistait pas à cette attaque [30].

5.1.2 Quelques notations

Nous utiliserons ici les mêmes notations que dans le chapitre 1. On considérera le corps $\mathbb{F}_q$ avec $q = 2^t$. L'ensemble des polynômes sur $\mathbb{F}_q$ de degré au plus k sera noté

$$\mathbb{F}_q^{(k)}[X].$$

Le code de Reed-Solomon de longueur $q-1$, de dimension k et de distance minimale $q-k$ sur l'alphabet $\mathbb{F}_q$ sera noté

$$RS[q-1,k].$$

On a défini la distance de Hamming et la distance de Hamming normalisée entre deux mots de codes. On a aussi vu que l'image d'une fonction $f : \mathbb{F}_q \to \mathbb{F}_q$ constituait un mot de code, donc par abus de notation on notera

$$d(f,RS[q-1,k]) = \min\{d(\mathrm{Im}(f),c) \mid c \in RS[q-1,k]\},$$

la plus petite distance de Hamming qui existe entre un mot du code de Reed-Solomon et l'image de f. On notera

$$\Delta(f,RS[q-1,k]) = \frac{d(f,RS[q-1,k])}{q-1}$$

la plus petite distance de Hamming normalisée qui existe entre un mot du code de Reed-Solomon et l'image de f. Soit $g : \mathbb{F}_q \to \mathbb{F}_q$. On note encore par abus de notation :

$$d(f,g)$$

la distance de Hamming des mots de $\mathbb{F}_q^{q-1}$, induite respectivement par les images de f et de g. On notera de même la distance de Hamming normalisée entre f et g :

$$\Delta(f,g) = \frac{d(f,g)}{q-1}.$$

Rappelons que ces distances peuvent être interprétées comme des probabilités, on peut aussi écrire :

$$\Delta(f,g) = \mathop{\mathbf{Prob}}_{x \in \mathbb{F}_q}[f(x) = g(x)],$$

où la probabilité est calculée sur une distribution uniforme des $x \in \mathbb{F}_q$.

5.2 Tests à clairs choisis

On va surtout voir ici qu'il est intéressant de pouvoir mesurer une propriété statistique à partir d'un échantillon, même corrélé. On va tout d'abord rappeler les résultats de M. Sudan qui sont applicables sur un corps premier [44]. Ces résultats seront ensuite étendus aux corps du type $\mathbb{F}_q$ avec $q = 2^t$ [48].

5.2.1 Tests sur un corps premier

On considère dans cette partie les corps du type $\mathbb{F}_p$ avec p premier. Ces résultats peuvent être trouvés dans la thèse de M. Sudan [44].

Définition 26. *On dit que les points* $\{x_0,\ldots,x_{k+1}\} \in \mathbb{F}_p^{k+1}$ *sont à espacements réguliers s'il existe* $h \in \mathbb{F}_p^*$ *tel que* $x_i = x_0 + i \cdot h$, *pour* $i \in \{0,\ldots,k+1\}$.

Lemme 2. *Soient k un entier positif, et un nombre premier $p \geq k+2$. Il existe un polynôme de degré au plus k passant par les points*

$$\{(x_i,y_i)|i \in \{0,\ldots,k+1\}; x_i = x + i \cdot h,\ \ x_i,y_i \in \mathbb{F}_p\}$$

si et seulement si $\sum_{i=0}^{k+1} \alpha_i y_i = 0$, où $\alpha_i = (-1)^{(i+1)}\binom{k+1}{i}$.

Preuve : On définit la suite de fonctions $f^{(j)}$, $j \in [1...k+1]$ telle que

$$f^{(0)}(x_i) = y_i,\ \ f^{(j)}(x_i) = f^{(j-1)}(x_i) - f^{(j-1)}(x_{i+1}),\ \ i \in [0,...,k+1].$$

Alors comme $p \geq k+2$, on peut utiliser la formule de Taylor pour les polynômes,

$$f^{(j-1)}(x_i) - f^{(j-1)}(x_i + h) = \sum_{l=1}^{j} \frac{h^l}{l!} D_l(f^{(j-1)})(x_i),$$

donc $f^{(j)}$ coïncide avec un polynôme de degré $k-j$ sur les $k+2$ éléments (x_i,y_i) si et seulement si $f^{(j-1)}$ coïncide avec un polynôme de degré $k-j+1$ sur ces mêmes éléments. Ceci implique que $f^{(k)}$ est constante et donc que $f^{(k+1)}(x_0) = 0$ si et seulement si $f^{(0)}$ est polynôme de degré k. Mais $f^{(k+1)}(x_0) = \sum_{i=0}^{k+1} \alpha_i y_i = 0$. □

Lemme 3. *Soient k un entier positif et p un nombre premier tel que $p \geq k+2$. La fonction $f : \mathbb{F}_p \to \mathbb{F}_p$ est un polynôme de degré au plus k si et seulement si*

$$\forall x,h \in \mathbb{F}_p,\ \sum_{i=0}^{k+1} \alpha_i f(x + i \cdot h) = 0,$$

où $\alpha_i = (-1)^{(i+1)}\binom{k+1}{i}$.

Preuve : Le lemme 2 implique que si f est un polynôme de degré au plus k, alors

$$\forall x,h \in \mathbb{F}_p,\ \sum_{i=0}^{k+1} \alpha_i f(x + i \cdot h) = 0.$$

Maintenant si

$$\forall x,h \in \mathbb{F}_p,\ \sum_{i=0}^{k+1} \alpha_i f(x + i \cdot h) = 0,$$

alors prenons par exemple $h = 1$. D'après le lemme 7, f est un polynôme de degré au plus k sur $x,x+1,...,x+k+1$. On peut itérer ce raisonnement : de même f est un polynôme de degré au plus k sur $x+1,x+2,...,x+k+2$. Par interpolation on en déduit que f est un polynôme de degré au plus k sur $x,x+1,x+2,...,x+k+2$ car $k+1$ couples (x_i,y_i) s'interpolent par un unique polynôme de degré k. De cette manière on recouvre l'ensemble $\mathbb{F}_p$, et on en déduit le lemme. □

Le théorème précédent montre qu'il suffit d'effectuer des interpolations sur des $(k+2)$-uplets $(x_i,f(x_i))$ où les x_i sont à espacements réguliers pour vérifier qu'une fonction est de bas degré.

Théorème 16. *[44] Soient k un entier positif, un nombre premier $p \geq k+2$, et une fonction $f : \mathbb{F}_p \rightarrow \mathbb{F}_p$ telle que*

$$\underset{x,h\in\mathbb{F}_p}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1}\alpha_i \cdot f(x+i\cdot h)=0\right] \geq 1-\delta \quad où \quad \delta \leq \frac{1}{2(k+2)^2},$$

avec $\alpha_i = (-1)^{(i+1)}\binom{k+1}{i}$, alors

$$\Delta(f,\mathbb{F}_p^k[X]) = \Delta(f,RS_p[p-1,k+1]) \leq 2\delta.$$

Le théorème 14 du chapitre 4 implique que pour $\delta \leq \frac{1}{2(k+2)^2}$, la taille de l'échantillon $\{x,h \in \mathbb{F}_p\}$ est de l'ordre de $\mathcal{O}(k^2)$. D'autre part, avec un précalcul des coefficients α_i, l'évaluation de l'expression $\sum_{i=0}^{k+1}\alpha_i \cdot f(x_i)$ coûte $k+1$ multiplications. Donc le coût du test qui résultera de ce théorème sera de l'ordre de $\mathcal{O}(k^3)$. La preuve du théorème précédent résulte des lemmes suivants :

Lemme 4. *Supposons que $\underset{x,h\in\mathbb{F}_p}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1}\alpha_i \cdot f(x+i\cdot h)=0\right] = 1-\delta$. Soit $g(x)$ le polynôme tel que la quantité*

$$\underset{x\in\mathbb{F}_p}{\mathbf{Prob}}\left[g(x)=\sum_{i=1}^{k+1}\alpha_i \cdot f(x+i\cdot h)\right]$$

soit maximale. Ce maximum est pris par rapport à h. Alors g et f coïncident sur une fraction d'au moins $1-2\delta$ des éléments de $\mathbb{F}_p$.

Preuve : Considérons l'ensemble des éléments $x \in \mathbb{F}_p$ tels que

$$\underset{h\in\mathbb{F}_p}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1}\alpha_i \cdot f(x+i\cdot h)=0\right] < 1/2.$$

Si la fraction de tels éléments est plus grande que 2δ alors cela contredit la condition que $\underset{x,h\in\mathbb{F}_p}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1}\alpha_i \cdot f(x_i)=0\right] = 1-\delta$. Donc, pour le reste des éléments, $f(x)=g(x)$. □

Lemme 5. *Pour tout $x \in \mathbb{F}_p$,*

$$\underset{h\in\mathbb{F}_p}{\mathbf{Prob}}\left[g(x)=\sum_{i=1}^{k+1}\alpha_i \cdot f(x+i\cdot h)=0\right] \geq 1-2(k+1)\delta.$$

Preuve : Observons que

$$h_1,h_2 \in \mathbb{F}_p \Rightarrow x+i\cdot h_1 \in \mathbb{F}_p \text{ et } x+j\cdot h_2 \in \mathbb{F}_p$$

$$\Rightarrow \underset{h_1,h_2}{\mathbf{Prob}}\left[f(x+i\cdot h_1)=\sum_{j=1}^{k+1}\alpha_j f(x+i\cdot h_1+j\cdot h_2)\right]\geq 1-\delta$$

et

$$\underset{h_1,h_2}{\mathbf{Prob}}\left[f(x+j\cdot h_2)=\sum_{i=1}^{k+1}\alpha_j f(x+i\cdot h_1+j\cdot h_2)\right]\geq 1-\delta,$$

en combinant ces deux expressions, on obtient

$$\underset{h_1,h_2}{\mathbf{Prob}}\left[\begin{array}{l}\sum_{i=1}^{k+1}\alpha_i f(x+i\cdot h_1)\\ =\sum_{i=1}^{k+1}\sum_{j=1}^{k+1}\alpha_i\alpha_j f(x+i\cdot h_1+j\cdot h_2)\\ =\sum_{j=1}^{k+1}\alpha_i f(x+j\cdot h_1)\end{array}\right]\geq 1-2(k+1)\delta$$

□

Lemme 6. *Pour tout $x,h \in \mathbb{F}_p$, si $\delta \leq \frac{1}{2(k+1)^2}$, alors $\sum_{i=1}^{k+1}\alpha_i g(x+i\cdot h)=0$ (et donc g est un polynôme de degré au plus k).*

Preuve : Soit $h_1,h_2 \in \mathbb{F}_p$. Alors $h_1+i\cdot h_2 \in \mathbb{F}_p$ implique que pour tout $0\geq i\geq k+1$

$$\underset{h_1,h_2}{\mathbf{Prob}}\left[g(x+i\cdot h)=\sum_{j=1}^{k+1}\alpha_j f((x+i\cdot h)+j\cdot(h_1+i\cdot h_2))\right]\geq 1-2(k+1)\delta.$$

De plus, on a pour tout $1\leq j\leq k+1$

$$\underset{h_1,h_2}{\mathbf{Prob}}\left[\sum_{i=1}^{k+1}\alpha_j f((x+j\cdot h_1)+i\cdot(h+j\cdot h_2))=0\right]\geq 1-\delta.$$

Ces deux expressions impliquent

$$\underset{h_1,h_2}{\mathbf{Prob}}\left[\begin{array}{l}\sum_{i=0}^{k+1}\alpha_i g(x+i\cdot h)=\\ \sum_{j=1}^{k+1}\sum_{i=0}^{k+1}\alpha_i\alpha_j f((x+j\cdot h_1)+i\cdot(h+j\cdot h_2))\\ =0\end{array}\right]\geq 1-2(k+1)^2\delta>0.$$

En particulier l'évènement "$\sum_{i=0}^{k+1}\alpha_i g(x+i\cdot h)=0$" ne dépend pas de h_1 et h_2, donc, comme cette probabilité est strictement positive, elle est forcément égale à 1. □

Maintenant, on peut déduire du théorème 16, un test de bas degré:

Théorème 17. *[44] Si les sorties d'une fonction f sont les images d'un polynôme de degré au plus k alors la fonction passera le test à espacements réguliers. Si les images d'une fonction sont telles que $\Delta(f,RS_p[p-1,k+1])>\frac{1}{(k+2)^2}$, alors avec probabilité $1-\beta$, elle sera rejetée par le test à espacements réguliers.*

```
Test-régulier := fonction(k,β,f)

N := O(k^2 log(1/β));
réponse := oui;

    pour i de 1 à N
    {
      Prendre x,h ∈ F_p × F_p
      si Σ_{i=0}^{k+1} α_i · f(x + i · h) ≠ 0
      {
        réponse := non;
        break;
      }
    }
retourner réponse;
```

FIG. 5.1 – *Test à espacements réguliers*

Preuve : Avec un probabilité de succès $1-\beta$ on peut tirer un couple $x,h \in \mathbb{F}_p$ tel que

$$\sum_{i=0}^{k+1} \alpha_i \cdot f(x+i\cdot h) \neq 0$$

si

$$\underset{x,h\in\mathbb{F}_p}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1} \alpha_i \cdot f(x_i) = 0\right] < 1 - \frac{1}{2(k+2)^2},$$

c'est une conséquence directe de la formule du théorème 14 du chapitre 4. Si

$$\underset{x,h\in\mathbb{F}_p}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1} \alpha_i \cdot f(x_i) = 0\right] < 1 - \frac{1}{2(k+2)^2}.$$

Alors le théorème 15 implique immédiatement que

$$\Delta(f,RS_p[p-1,k+1]) \leq \frac{1}{(k+2)^2}.$$

□

5.2.2 Extension des Tests aux corps binaires étendus

Les tests précédents ne sont malheureusement pas applicables directement pour les corps de type $\mathbb{F}_q$ avec $q = 2^t$. On se limitera à une étude en caractéristique 2, car c'est la caractéristique la plus courante dans le domaine des télécommunications et de la cryptographie à clef secrète. On va donc étendre ces résultats à la caractéristique 2. On notera ω un élément primitif de $\mathbb{F}_{2^t}$.

Définition 27. *On dira que les éléments* $\{x_0,\ldots,x_{k+1}\} \in \mathbb{F}_q^{k+2}$ *sont à espacements réguliers s'il existe* $x,h,\omega \in \mathbb{F}_q \times \mathbb{F}_q^* \times \mathbb{F}_q^*$, *tels que* $x_0 = x$ *et* $x_i = x + h\cdot\omega^{i-1}$ *pour* $i \in \{1,\ldots,k+1\}$ *(ω un élément primitif de $\mathbb{F}_{2^t}$).*

Proposition 2. *Soient* $\{x_0,x_1,\ldots,x_{k+1}\}$ $k+2$ *éléments distincts de* $\mathbb{F}_q$, *et une fonction* $f : \mathbb{F}_q \to \mathbb{F}_q$ *telle que* $f(x_i) = y_i$ *pour* $i \in \{0,\ldots,k+1\}$. *Soit la suite de fonctions* $f^{(j)}$ *construite comme il suit : pour* $j \in \{1,\ldots,k+1\}$ *et* $s \in \{1,\ldots,k-j+2\}$, *on pose*

$$f^{(j)}(x_{s-1},\ldots,x_{s+j-1}) = \frac{f^{(j-1)}(x_{s-1},\ldots,x_{s+j-2}) - f^{(j-1)}(x_s,\ldots,x_{s+j-1})}{x_{s-1} - x_{s+j-1}}, \quad (5.1)$$

$$f^{(0)}(x_i) = f(x_i) = y_i, \quad i \in \{0,\ldots,k+1\}, \quad (5.2)$$

alors l'ensemble $\{(x_0,f(x_0)),\ldots,(x_{k+1},f(x_{k+1}))\}$ *est interpolable par un polynôme de degré au plus k si et seulement si* $f^{(k+1)}(x_0,\ldots,x_{k+1}) = 0$. *D'autre part, si* $f(x)$ *est un polynôme de degré k avec* $f(x) = \sum_{i=0}^{k} c_i x^i$, *alors on a*

$$c_k = f^{(k)}(x_0,\ldots,x_k) = f^{(k)}(x_1,\ldots,x_{k+1}). \quad (5.3)$$

Preuve : Par définition, f est un polynôme de la forme $f = \sum_i c_i X^i$. On va calculer l'expression $f^{(k+1)}(x_0,\ldots,x_{k+1})$. Par linéarité de l'expression 5.1, on peut se restreindre au calcul de $f^{(k+1)}$ pour un monôme X^T. Considérons $k+2$ variables $x_0,\ldots,x_{k+1}$. On a donc $f^{(0)}(x_i) = x_i^T$. On raisonne par récurrence : si $j = 1$ on voit que

$$f^{(1)}(x_{s-1},x_s) = \frac{x_{s-1}^T - x_s^T}{x_{s-1} - x_s},$$

est la somme de tous les monômes en x_{s-1},x_s ($s \in \{1,\ldots,k-j+2\}$) de degré $T-1$.

Au rang j, supposons que $f^{(j)}(x_{s-1},\ldots,x_{s+j-1})$ est la somme de tous les monômes de degré $t-j$. Maintenant au rang $j+1$, pour tout monôme $x_{s-1}^{i_{s-1}}\ldots x_{s+j-1}^{i_{s+j-1}}$ de $f^{(j)}(x_{s-1},\ldots,x_{s+j-1})$ on a le monôme $x_{s+j}^{i_{s-1}}\ldots x_{s+j-1}^{i_{s+j-1}}$ de $f^{(j)}(x_{s-1},\ldots,x_{s+j-1})$ avec $i_{s-1} + \ldots + i_{s+j-1} = T - j$, et

$$x_{s-1}^{i_{s-1}} x_s^{i_s} \ldots x_{s+j-1}^{i_{s+j-1}} - x_{s+j}^{i_{s-1}} x_s^{i_s} \ldots x_{s+j-1}^{i_{s+j-1}} = (x_{s-1} - x_{s+j}) \cdot M_{s-1,s+j}^{i_{s-1}-1} \cdot x_s^{i_s} \ldots x_{s+j-1}^{i_{s+j-1}}$$

où $M_{s-1,s+j}^{i_{s-1}-1}$ est la somme de tous les monômes de degré $i_{s-1}-1$ en x_{s-1},x_{s+j}, donc on voit que $M_{s-1,s+j}^{i_{s-1}-1} \cdot x_s^{i_s} \ldots x_{s+j-1}^{i_{s+j-1}}$ est la somme des monômes de degré $t-j-1$

en $x_s,\ldots,x_{s+j-1}$. On obtient que $f^{(j+1)}$ est la somme de tous les monômes de degré $T-j-1$.

La récurrence est montrée, donc, en particulier, $f^{(k+1)}(x_0,\ldots,x_{k+1})$ est la somme de tous les monômes de degré $T-k-1$. Donc en particulier si l'ensemble $\{(x_0,f(x_0)),\ldots,(x_{k+1},f(x_{k+1}))\}$ avec $\{x_0,\ldots,x_{k+1}\}\in\mathbb{F}_q$ est interpolable par un polynôme de degré au plus k, alors $f^{(k+1)}(x_0,\ldots,x_{k+1})=0$.

Réciproquement, si

$$f^{(k+1)}(x_0,\ldots,x_{k+1})=0,$$

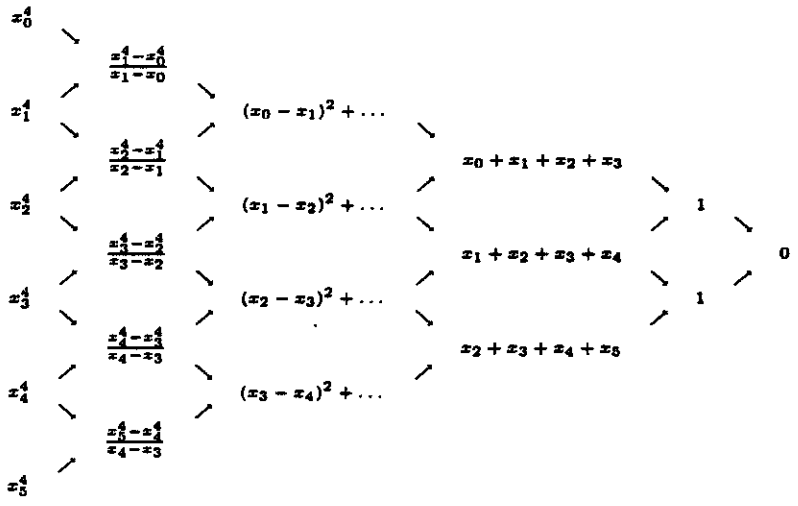
supposons alors que f soit interpolable par un polynôme P de degré $k+1$ sur $\{x_0,\ldots,x_{k+1}\}\in\mathbb{F}_q$: c'est le pire cas qui puisse être considéré puisque sinon la propriété est montrée. Posons

$$P(X)=\sum_{i=0}^{k+1}c_iX^i=h(X)+c_{k+1}X^{k+1}.$$

Définissons la suite de fonctions de la proposition associée à P. Alors comme h est de degré au plus k, on a $h^{(k+1)}(x_0,\ldots,x_{k+1})=0$. D'autre part, on a montré que pour $f(X)=X^{k+1}$, $f^{(k+1)}(x_0,\ldots,x_{k+1})$ est la somme de tous les monômes de degré 0, c'est-à-dire 1. Donc en conclusion $P^{(k+1)}(x_0,\ldots,x_{k+1})=c_{k+1}=0$ et la première assertion de la proposition est montrée. La deuxième vient d'être montrée, par linéarité de l'expression 5.1, en effet, si $f(x)=\sum_{i=0}^{k}c_ix^i$ est un polynôme de degré k alors on a $c_k=f^{(k)}(x_0,\ldots,x_k)$. □

Exemple 14 :

Soit $f(x)=x^4$. On fait un test pour $k=4$. On va donc représenter le calcul des fonctions $f^{(j)}$, et vérifier que $f^{(k+1)}=0$. On a pris 6 éléments distincts de $\mathbb{F}_q$ $x_0,\ldots,x_5$. Voici une figure qui illustre les opérations que l'on effectue:



◊

Théorème 18. *Soient k,t des entiers tels que* $2^t > k+1$. *Soit* $f : \mathbb{F}_{2^t} \to \mathbb{F}_{2^t}$ *une fonction. Soit* $\{x_0,\ldots,x_{k+1}\}$ *un ensemble de points à espacement régulier avec* $x_0 = x$ *et* $x_i = x + h \cdot \omega^{i-1}$. *Soit* $y_i = f(x_i)$, $i \in \{0,\ldots,k+1\}$. *L'ensemble des* (x_i,y_i) *sont les entrées et sorties d'un polynôme de degré au plus k si et seulement si* $\sum_{i=0}^{k+1} \alpha_i(\omega,k) \cdot y_i = 0$ *où les* α_i *sont donnés par les expressions suivantes:*

$$B_1^0 = B_1^1 = 1,\ B_i^0 = B_{i-1}^0/\omega^{k-i+2},\ B_i^i = B_{i-1}^{i-1}/(\omega^{i-1} - \omega^k)$$

et

$$B_i^s = B_{i-1}^{s-1}/A_{i-1}^{s-1} - B_{i-1}^s/A_{i-1}^s,\ \text{où}\ A_i^s = \omega^{s-1} - \omega^{s+k-i},\ s \in \{1,\ldots,i-1\}$$

avec $\alpha_s(\omega,k) = B_{k+1}^s$.

Preuve : C'est une conséquence directe de la proposition 2. Rappelons que l'on peut construire la suite de fonctions $f^{(j)}$ suivante: pour $j \in \{1,\ldots,k+1\}$ et $s \in \{1,\ldots,k-j+2\}$, on pose

$$f^{(j)}(x_{s-1},\ldots,x_{s+j-1}) = \frac{f^{(j-1)}(x_{s-1},\ldots,x_{s+j-2}) - f^{(j-1)}(x_s,\ldots,x_{s+j-1})}{x_{s-1} - x_{s+j-1}},$$

$$f^{(0)}(x_i) = f(x_i) = y_i,\ \ i \in \{0,\ldots,k+1\},$$

alors l'ensemble $\{(x_0,f(x_0)),\ldots,(x_{k+1},f(x_{k+1}))\}$ est interpolable par un polynôme de degré au plus k si et seulement si $f^{(k+1)}(x_0,\ldots,x_{k+1}) = 0$. Il suffit donc maintenant de donner $f^{(k+1)}(x_0,\ldots,x_{k+1})$ en fonction de h,ω. Il s'agit donc de montrer par récurrence que

$$f^{(k+1)}(x_0,\ldots,x_{k+1}) = \frac{1}{h^j}\sum_{s=0}^{j} \alpha_t(\omega,j-1) f^{(k+1-j)}(x_s,\ldots,x_{s+k+1-j}).$$

Suposons que cette propriété est vraie jusqu'au rang k. Par définition, on a

$$f^{(k+1-j)}(x_s,\ldots,x_{s+k+1-j}) = \frac{f^{(k-j)}(x_s,\ldots,x_{s+k-j}) - f^{(k-j)}(x_{s+1},\ldots,x_{s+k-j+1})}{x_s - x_{s+k-j+1}},$$

avec $x_s - x_{s+k-j+1} = h \cdot (\omega^s - \omega^{s+k-j+1})$. En conséquence, on obtient l'expression

$$f^{(k+1)}(x_0,\ldots,x_{k+1}) = \frac{1}{h^{j+1}}\sum_{s=0}^{j+1} \beta_s(\omega,j) f^{(k-j)}(x_s,\ldots,x_{s+k-j}),$$

où

$$\beta_s(\omega,j) = \frac{\alpha_{s-1}(\omega,j-1)}{\omega^{s-1} - \omega^{s+k-j}} - \frac{\alpha_s(\omega,j-1)}{\omega^s - \omega^{s+1+k-j}} = \alpha_s(\omega,j),$$

et donc la propriété est démontrée. □

Remarque 10. *Ce dernier théorème est très utile car on peut pré-calculer les coefficients $\alpha_s(\omega,k)$. La complexité de ce pré-calcul est en $\mathcal{O}(k^2)$ opérations arithmétiques dans le corps $\mathbb{F}_{2^t}$. La méthode pour pré-calculer ces coefficients est exactement la même que pour calculer les coefficients binomiaux, c'est le triangle de Pascal. Une fois ces coefficients pré-calculés, le coût du calcul de $\sum_{i=0}^{k+1} \alpha_i(\omega,k) \cdot y_i$ ne dépend plus que de la multiplication sur $\mathbb{F}_{2^t}$, c'est-à-dire $\mathcal{O}(k)$.*

Lemme 7. *Soient k,t des entiers positifs tels que $2^t \geq k+2$. La fonction $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$, $q = 2^t$ est un polynôme de degré au plus k si et seulement si*

$$\forall x,h \in \mathbb{F}_p, \sum_{i=0}^{k+1} \alpha_i f(x + h \cdot \omega^{i-1}) = 0,$$

où les α_i sont construits comme dans la proposition précédente.

Preuve : Le théorème 18 implique que si f est un polynôme de degré au plus k, alors

$$\forall x,h \in \mathbb{F}_q \times \mathbb{F}_q^*, \sum_{i=0}^{k+1} \alpha_i f(x + h \cdot \omega^{i-1}) = 0.$$

Maintenant, si $\forall x,h \in \mathbb{F}_q \times \mathbb{F}_q^*$, $\sum_{i=0}^{k+1} \alpha_i f(x + h \cdot \omega^{i-1}) = 0$, alors prenons par exemple $h = 1$ et $x = 0$. Alors d'après le théorème 16, f est un polynôme de degré au plus k sur $1,\omega,...,\omega^k$. On peut itérer ce raisonnement. De même en prenant $x = 0$ et $h = \omega$, f est un polynôme de degré au plus k sur $\omega,...,\omega^{k+1}$. Par interpolation, on en déduit que f est un polynôme de degré au plus k sur $1,\omega,...,\omega^{k+1}$. De cette manière, on recouvre l'ensemble $\mathbb{F}_q$, et on en déduit le lemme. □

Le théorème précédent montre qu'il suffit d'effectuer des interpolations sur des $(k+2)$-uplets $(x_i,f(x_i))$ où les x_i sont à espacements réguliers pour vérifier qu'une fonction est de bas degré.

Théorème 19. *Soient k,t des entiers positifs tels que $2^t \geq k+2$, et une fonction $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$, $q = 2^t$ telle que*

$$\underset{x,h \in \mathbb{F}_q \times \mathbb{F}_q^*}{\mathbf{Prob}} \left[\sum_{i=0}^{k+1} \alpha_i \cdot f(x_i) = 0 \right] \geq 1 - \delta \quad \text{où} \quad \delta \leq \frac{1}{2(k+2)^2},$$

avec α_i construit comme dans le théorème 18, et où la probabilité est prise sur des $(k+2)$-uplets à espacements réguliers. Alors $\Delta(f,RS_q[q-1,k+1]) \leq 2\delta$.

Preuve : C'est exactement la même preuve que celle du théorème 16. Il suffit simplement de remplacer les expression du type $i \cdot h$ par $h \cdot \omega^{i-1}$. □

Maintenant, on peut déduire du théorème 19 un test de bas degré :

Théorème 20. *Si les sorties d'une fonction f sont les images d'un polynôme de degré au plus k alors la fonction passera le test à espacements réguliers. Si les images d'une fonction sont telles que $\Delta(f,RS_q[q-1,k+1]) > \frac{1}{(k+2)^2}$, alors avec probabilité $1-\beta$, elle sera rejetée par le test à espacements réguliers (2).*

```
Test-régulier := fonction(k,β,f)

N := O(k^2 log(1/β));
réponse := oui;

    pour i de 1 à N
    {
      Prendre x,h ∈ F_q × F_q^*
      si Σ_{i=0}^{k+1} α_i · f(x + i · h) ≠ 0
      {
         réponse := non;
         break;
      }
    }
retourner réponse;
```

FIG. 5.2 – *Test à espacements réguliers (2)*

Preuve : Avec un probabilité de succès $1-\beta$, on peut tirer un couple

$$x,h \in \mathbb{F}_q \times \mathbb{F}_q^*$$

tel que

$$\sum_{i=0}^{k+1} \alpha_i \cdot f(x+i\cdot h) \neq 0$$

si

$$\underset{x,h\in\mathbb{F}_q\times\mathbb{F}_q^*}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1}\alpha_i\cdot f(x_i)=0\right] < 1-\frac{1}{2(k+2)^2};$$

c'est une conséquence direct su théorème 14 du chapitre 4. Si

$$\underset{x,h\in\mathbb{F}_q\times\mathbb{F}_q^*}{\mathbf{Prob}}\left[\sum_{i=0}^{k+1}\alpha_i\cdot f(x_i)=0\right] < 1-\frac{1}{2(k+2)^2},$$

alors le théorème 14 implique immédiatement que

$$\Delta(f,RS_q[q-1,k+1]) \leq \frac{1}{(k+2)^2}.$$

□

Conclusion

Ces tests sont très pratiques pour tester le degré d'une fonction. Ils révèlent que tester le degré d'une fonction sur un échantillon fortement corrélé suffit à caractériser cette fonction sur l'ensemble de son domaine de définition. Prendre des points à espacement régulier est pratique puisque le calcul des coefficients α_i est un pré-calcul. Une fois ce pré-calcul effectué, il ne reste qu'a faire des multiplication dans le corps considéré pour effectuer ces tests. Cependant ces tests restent faibles pour les problèmes cryptographiques qui nous intéressent. Nous nous intéresserons à des problèmes similaires, mais dans le contexte multivarié, dans le prochain chapitre.

5.3 Tests à clairs connus

Considérons une fonction $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$. On va tout d'abord considérer le problème "Poly-agree", c'est-à-dire que l'on se demande s'il existe au moins un polynôme p de degré au plus k tel que $\Delta(f,p) \leq \delta$. En terme d'information, répondre à ce problème est moins fort que de répondre au problème de reconstruction; en effet si on est capable de reconstruire le polynôme p de degré au plus k le plus proche de f, alors par simple échantillonage on obtient la réponse au problème "Poly-agree". Les tests que nous allons étudier sont basés sur le problème d'interpolation. En effet, on sait que par $k+1$ couples d'éléments de $\mathbb{F}_q$ il passe forcément un polynôme de degré au plus k, et cette propriété est en fait vraie sur n'importe quel corps. Donc pour mesurer une telle propriété algébrique, il faut considérer au moins $k+2$ couples.

5.3.1 Un test simple

Nous allons décrire un test simple qui est appelé *"Basic Univariate Test"* [44], ainsi qu'un test qui est simplement suggéré dans [44].

Lemme 8. *Soient le corps $\mathbb{F}_q$, $(q = 2^t)$, k un entier tel que q soit très grand devant k. Soit une fonction $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$. Alors si*

$$\Delta(f,RS[q-1,k+1]) = \delta,$$

on a

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q^2}{\mathbf{Prob}} \left[\exists g \in \mathbb{F}_q^{(k)}[X] \mid g(x_i) = f(x_i) \ \ \forall i \in [0\ldots k+1]\right] \geq (1-\delta)^{k+2},$$

où la probabilité est prise sur la distribution uniforme des $k+2$ uplets. Soit maintenant un polynôme $g \in \mathbb{F}_q^{(k)}[X]$ qui minore la quantité $d(f,g)$, alors on a:

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q^2}{\mathbf{Prob}} \left[g(x_i) = f(x_i) \ \ \forall i \in [0\ldots k+1]\right] = (1-\delta)^{k+2},$$

où la probabilité est prise sur la distribution uniforme des $k+2$ uplets.

Test := fonction(k,f) :

Prendre aléatoirement $k+2$ points distincts $x_0,\ldots,x_k,x_{k+1} \in \mathbb{F}_q$

Si f s'interpole sur $(x_0,f(x_0)),\ldots,(x_{k+1},f(x_{k+1}))$
par un polynôme de degré au plus k, alors
réponse := oui;
Sinon
réponse := non;

retourner réponse;

FIG. 5.3 – *"Basic Univariate Test"*

Preuve : Supposons que $\Delta(f,RS[q-1,k+1]) = \delta$. Soit, alors, un élément $g \in \mathbb{F}_q^{(k)}[X]$ tel que $\delta = \Delta(f,g)$. On sait alors qu'il y a $q-\delta q$ éléments $x_i \in \mathbb{F}_q$ qui vérifient $f(x) = g(x)$, donc si δ est proche de 0, la probabilité de tirer $k+2$ tels éléments est égale à

$$\frac{\binom{q-\delta q}{k+2}}{\binom{q}{k+2}} \approx (1-\delta)^{k+2},$$

on en déduit donc la deuxième assertion du lemme :

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q}{\mathbf{Prob}} [g(x_i) = f(x_i) \ \ \forall i \in [0\ldots k+1]] = (1-\delta)^{k+2}.$$

La première assertion provient du fait qu'il peut y avoir $k+2$ couples interpolables par un autre polynôme que g; on en déduit donc :

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q}{\mathbf{Prob}} \left[\exists g \in \mathbb{F}_q^{(k)}[X] \mid g(x_i) = f(x_i) \ \ \forall i \in [0\ldots k+1]\right] \geq (1-\delta)^{k+2},$$

où la probabilité est prise sur la distribution uniforme des $k+2$ uplets. □

Lemme 9. *[44] Soit k un entier, $\mathbb{F}_q$ un corps fini de dimension supérieure à que $k+2$ et une fonction $f : \mathbb{F}_q \to \mathbb{F}_q$; si f satisfait*

$$\underset{x_i\in\mathbb{F}_q}{\mathbf{Prob}} \left[\exists g \in \mathbb{F}_q^{(k)}[x] \text{ tel que } g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right] \geq 1-\delta,$$

où la probabilité est prise sur la distribution uniforme des $k+2$ uplets $\{x_0,x_1,\ldots,x_{k+1}\}$ d'éléments distincts de $\mathbb{F}_q$, alors $\Delta(f,RS[q-1,k+1]) \leq \delta$.

Preuve : Soit $\delta' = \Delta(f,RS[q-1,k+1])$. Maintenant on fixe $z_0,\ldots,z_k$. Soit h l'unique polynôme de degré k tel que $h(z_i) = f(z_i)$, pour $i \in \{0,\ldots,k\}$. D'après

la définition de δ', on a que $\underset{x_{k+1}\in\mathbb{F}_q}{\mathbf{Prob}}\,[h(x_{k+1}) = f(x_{k+1})] \leq 1-\delta'$, donc

$$\underset{x_0,\ldots,x_{k+1}}{\mathbf{Prob}}\left[\exists P \in \mathbb{F}_q^{(k)}[x]) \text{ tel que } \forall i \in [0,\ldots,k+1],\ P(x_i) = f(x_i)\right]$$

$$\leq \max_{z_0,\ldots,z_k} \underset{x_{k+1}}{\mathbf{Prob}}[\exists P \in \mathbb{F}_q^{(k)}[x]) \text{ tel que } \forall i \in [0,\ldots,k],\ P(z_i) = f(z_i) \mid$$

$$P(x_{k+1}) = f(x_{k+1})],$$

$$\leq 1-\delta'.$$

□

Les lemmes précédents suggèrent un test qui passe tout d'abord par une évaluation de la quantité :

$$\underset{x_i\in\mathbb{F}_q}{\mathbf{Prob}}\left[\exists g \in \mathbb{F}_q^{(k)}[x] \text{ tel que } g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right].$$

Le programme de la figure 5.4 va estimer cette quantité, à condition que la taille de l'échantillon soit correctement choisie.

```
Estime := fonction(k,f,N) :

compteur := 0;

Répéter N fois
{
      Prendre k+2 points distincts x_0,...,x_k,x_{k+1} ∈ F_q

      Si f s'interpole sur (x_0,f(x_0)),...,(x_{k+1},f(x_{k+1}))
                        par un polynôme de degré au plus k, alors
        compteur := compteur + 1;
}

retourner compteur / N;
```

FIG. 5.4 – *Programme d'estimation*

Par exemple, supposons que

$$\underset{x_i\in\mathbb{F}_q}{\mathbf{Prob}}\left[\exists g \in \mathbb{F}_q^{(k)}[x] \text{ tel que } g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right] = 1-\delta.$$

Alors d'après le théorème 14 du chapitre 4, si l'on veut approcher cette probabilité $1-\delta$ avec une précision α, et avec une probabilité de succès au moins égale à $1-S$ Alors une taille d'échantillon de l'ordre de

$$N = \mathcal{O}\left(\alpha^{-2}\log(1/S)\right)$$

conviendra. On peut donc maintenant donner le test qui est suggéré dans la thèse de M. Sudan, et que l'on a représenté la figure 5.5.

```
Testeur := fonction(f,k,δ)

    s := 0;
    N := O(δ^-2 log(1/β))
    réponse := oui;
    pour i de 1 à N faire
      {
        prendre k+2 points aléatoires {x_0,...,x_{k+1}} dans F_q

        si (f^(k+1)(x_0,...,x_{k+1}) ≠ 0) alors
          {
            réponse := non
            break;
          }
      }

  retourner réponse;
```

FIG. 5.5 – *Testeur basique univarié*

Lemme 10. *Si les sorties d'une fonction sont les images d'un polynôme de degré au plus k, alors la fonction passera l'épreuve du testeur de la figure 5.5. Si les images d'une fonction sont telles que $\Delta(f,RS[q-1,k+1]) > \delta$, alors avec probabilité $1-\beta$, elle sera rejeté par le testeur de la figure 5.5.*

Preuve : Avec une probabilité de succès $1-\beta$ on peut tirer un $k+2$ uplet d'éléments de $\mathbb{F}_q$ tel que f soit interpolable sur ce $k+2$ uplet si

$$\underset{x_i \in \mathbb{F}_q}{\mathbf{Prob}}\left[\exists g \in \mathbb{F}_q^{(k)}[x] \text{ tel que } g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right] > 1-\delta,$$

c'est exactement ce que dit le théorème 14 du chapitre 4. Si

$$\underset{x_i \in \mathbb{F}_q}{\mathbf{Prob}}\left[\exists g \in \mathbb{F}_q^{(k)}[x] \text{ tel que } g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right] > 1-\delta.$$

Alors le lemme 3 implique que

$$\Delta(f,RS[q-1,k+1]) \leq \delta.$$

□

5.3.2 Remarques générales

D'après le lemme 8 si $\Delta(f,RS[q-1,k+1]) \leq \delta$ et si δ est petit devant $1/k$, alors

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q}{\mathbf{Prob}} \left[\exists g \in \mathbb{F}_q^{(k)}[X] \mid g(x_i) = f(x_i) \ \ \forall i \in [0 \ldots k+1]\right] \geq 1-(k+2)\delta.$$

D'autre part si

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q}{\mathbf{Prob}} \left[\exists g \in \mathbb{F}_q^{(k)}[X] \mid g(x_i) = f(x_i) \ \ \forall i \in [0 \ldots k+1]\right] \geq 1-(k+2)\delta,$$

alors le lemme 9 permet seulement de dire que

$$\Delta(f,RS[q-1,k+1]) \leq (k+2)\delta.$$

Donc le lemme 9 donné ne permet pas de donner un algorithme qui donnerait une estimation de la distance

$$\Delta(f,RS[q-1,k+1]).$$

Le lemme 8 semble ne pas nous donner non plus d'information précise sur cet estimation puisque si $\Delta(f,RS[q-1,k+1]) \leq \delta$, alors on qu'une majoration de la quantité

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q}{\mathbf{Prob}} \left[\exists g \in \mathbb{F}_q^{(k)}[X] \mid g(x_i) = f(x_i) \ \ \forall i \in [0 \ldots k+1]\right]$$

par $(1-\delta)^{k+2}$. Cependant on verra qu'espérimentalement que l'estimation de cette probabilité nous donne une information précise sur $\Delta(f,RS[q-1,k+1])$, simplement on ne sait pas le démontrer mathématiquement. Cette estimation de la distance reste de toute manière bien faible par rapport au résultats de M. Sudan. [45] (cf. section 7 chapitre 1). En effet pour estimer une probabilité égale à $(1-\delta)^{k+2}$ avec une précision α et une probabilité de succès égale à $1-S$, il faut d'après la loi centrale limite (cf. chapitre 4 théorème 14) un échantillon de l'ordre de

$$N = \mathcal{O}\left((1-\delta)^{k+2}(1-(1-\delta)^{k+2})\alpha^{-2}\log(1/S)\right).$$

Donc le coût de cette estimation serait exponentiel en le degré k que l'on veut tester. Rappelons que si l'on veut reconstruire les polynômes P satisfaisant

$$\Delta(f,P) \leq \delta,$$

alors le théorème de Sudan [45] nous dit qu'un échantillon de taille $2k/(1-\delta)^2$ est suffisant et que la complexité pour effectuer cette reconstruction est de l'ordre de

$$O\left(\frac{(2k)^2}{(1-\delta)^4}\log\left(\frac{2k}{(1-\delta)^2}\right)\right).$$

Une fois les polynômes P reconstruits, on peut estimer par un simple échantillonnage la probabilité de coïncidence avec la fonction f, et cela donne immédiatement la distance $\Delta(f,P)$. De cette manière le coût d'une estimation de la distance $\Delta(f,RS[q-1,k+1]$ reste polynomiale en le degré k.

5.4 La reconstruction

On va étudier ici différents algorithmes qui permettent d'effectuer le test d'interpolation de la figure 5.5.

5.4.1 L'interpolation

On veut donc construire un algorithme qui détermine si un $(k+2)$-uplet de couples $(x_i,f(x_i))$ est interpolable par un polynôme de degré au plus k. La méthode naturelle est d'effectuer une interpolation sur $k+1$ premiers couples $(x_i,f(x_i))$, puis d'effectuer une évaluation sur le dernier élément x_{k+1}. Avec une interpolation de Newton [10], le coût de cette méthode est de $\frac{5}{2}k^2+k$ opérations arithmétiques dans le corps $\mathbb{F}_q$, en négligeant le coût des additions et soustractions devant celui des multiplications. Cependant, si l'on conserve les polynômes que l'on a obtenus par interpolation, et que l'on conserve ceux qui apparaissent le plus grand nombres de fois, on ne fait rien d'autre que de la reconstruction. On a représenté à la figure 5.6 un programme simple capable de faire ce travail

D'après le théorème 15 du chapitre 4, si $\Delta(f,RS[q-1,k+1]) = \delta$, alors on doit estimer une probabilité égale à $(1-\delta)^{k+2}$ (cf. lemme 8) avec une précision α et une probabilité de succès égale à $1-S$, il faut d'après la loi centrale limite (cf. chapitre 4 théorème 14) un échantillon de l'ordre de

$$N = \mathcal{O}\left(\alpha^{-2}\log(1/S)\right).$$

Le coût de cette reconstruction est bien sûr exponentiel en le degré k et si ce programme est beaucoup moins efficace que l'algorithme de Sudan, il est important de constater que l'on a fait les mêmes calculs que pour le testeur de la figure 5.5.

5.4.2 Un autre type d'interpolation

Rappelons que, d'après la proposition 2, si les $k+2$ couples

$$(x_0,f(x_0)),\ldots,(x_{k+1},f(x_{k+1})),$$

Construct := fonction(k,f,δ,α,S) :

compteur := 0;
$N := N = \mathcal{O}\left((1-\delta)^{k+2}(1-(1-\delta)^{k+2})\alpha^{-2}\log(1/S)\right)$;
$L :=$ (liste de polynômes)
Répéter N fois
{
Prendre $k+2$ points distincts $x_0,\ldots,x_k,x_{k+1} \in \mathbb{F}_q$

Si f s'interpole sur $(x_0,f(x_0)),\ldots,(x_{k+1},f(x_{k+1}))$
par un polynôme $P(x)$ de degré au plus k, alors
$L := L \cup P(x)$;
}

Trier la liste L selon la fréquence d'apparition des polynômes;
retourner la liste de polynômes qui apparaissent plus de
$((1-\delta)^{k+2} - \alpha)N$ *fois*;

FIG. 5.6 – *Reconstruction par interpolation*

sont interpolables par un polynôme $P(x) = \sum_{i=0}^{k} c_i x_i$ de degré au plus k, alors on a :

$$c_k = f^{(k)}(x_0,\ldots,x_k) = f^{(k)}(x_1,\ldots,x_{k+1}),$$

où $f^{(j)}$ est construite comme dans la proposition 2. On va donc utiliser cette propriété pour reconstruire un polynôme P qui satisfait

$$\Delta(f,RS[q-1,k+1]) = \Delta(f,P) = \delta.$$

On rappelle donc que l'on a

$$\underset{(x_i,f(x_i))\in\mathbb{F}_q}{\mathbf{Prob}} [P(x_i) = f(x_i) \ \ \forall i \in [0\ldots k+1]] = (1-\delta)^{k+2},$$

où la probabilité est prise sur la distribution uniforme des $k+2$ uplets. Donc le nombre de tirages à effectuer pour obtenir une fraction de c_k de l'ordre $(1-\delta)^{k+2}$ est d'après le théorème 14 du chapitre 4 de l'ordre de

$$N = \mathcal{O}\left(\alpha^{-2}\log(1/S)\right),$$

avec probabilité de succès au moins égale à $1-S$ pour un écart-type de l'erreur de α. On détermine ainsi les coefficient c_k qui ont une chance d'être le coefficient d'une solution à notre problème de reconstruction. Comme les polynômes que

nous recherchons sont de degré au plus k, on construira un algorithme qui comportera $k+1$ étapes. Ces $k+1$ étapes correspondront à la détermination des $k+1$ coefficients de P. On vient de voir que l'on pouvait obtenir le dernier coefficient c_k 5.3 de P en tirant aléatoirement

$$\mathcal{O}\left(\alpha^{-2}\log(1/S)\right)$$

éléments de $\mathbb{F}_q$, de la même manière que dans le programme 5.5 Maintenant supposons que l'on ait déterminé un dernier coefficient c_k, alors on a

$$\Delta(f - c_k X^k, RS[q-1,k]) \leq \delta.$$

On a donc diminué le degré du polynôme à reconstruire, en l'occurence P.

Lemme 11. *Soit $f : \mathbb{F}_q \to \mathbb{F}_q$ une fonction. Supposons que*

$$\Delta(f, RS[q-1,k+1]) = \delta < \frac{1}{2}.$$

Alors on peut reconstruire l'unique polynôme $P \in \mathbb{F}_q^{(k)}[X]$ qui vérifie $\Delta(f,P) = \delta$ à l'aide du programme de la figure 5.7 en

$$\mathcal{O}\left(\alpha^{-2}\log(1/S)\right),$$

opération arithmétiques sur le corps $\mathbb{F}_q$, avec une probabilité de succès au moins égale à $1 - S$ (avec un écart-type de l'erreur de α).

Preuve : On a vu dans la preuve de la proposition 2 que le coût d'une vérification d'interpolation sur un $k+2$ uplet $(x_i, f(x_i))$ était de $\frac{(k+1)(k+2)}{2}$ divisions sur $\mathbb{F}_q$. On a $k+1$ étapes à effectuer où le degré est abaissé de 1 à chaque étape. D'autre part, à chaque étape on effectue $\mathcal{O}\left(\alpha^{-2}\log(1/S)\right)$ tirages. Comme $\delta < 1/2$ il y a un unique candidat P qui répond au problème, on en déduit donc la complexité finale. □

5.4.3 Résultats expérimentaux

On a vu, par les lemmes 8 et 9, qu'une bonne approximation de

$$\mathbf{Prob}\left[\exists g \in \mathbb{F}_q^{(k)}[x] \text{ tel que } g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right],$$

ne donne pas forcément une bonne approximation de la distance $\Delta(f, RS[q-1,k+1])$, ces lemmes ne donnent qu'une majoration ou une minoration de la valeur de $\Delta(f, RS[q-1,k+1])$, mais qu'en est-il dans la pratique? Pour répondre à cette question, on a construit la fonction $f : \mathbb{F}_q \to \mathbb{F}_q$, $q = 2^{64}$ telle que

$$f(x) = x^4 + x^2 + 1 \text{ avec probabilité } 1 - \delta.$$

Basic-reconstructeur := fonction(f,k,δ,S,α)

$N := \mathcal{O}(\alpha^{-2}\log(1/S))$;
$M := [0]$; $M' := \{\}$;
$M'' := []$; $g := f$;

Pour j := 0 jusqu'à k faire
{
si card(M) = 0
{
imprimer : "Echec de la reconstruction";
break;
}
pour l de 1 à card(M) faire
{
$g = f - M[l]$;
pour i de 1 à N faire
{
prendre $k+2-j$ éléments aléatoires $\{x_0,\ldots,x_{k+1}\} \in \mathbb{F}_q^{k+2-j}$
si $(g^{(k+1-j)}(x_0,\ldots,x_{k+1-j}) = 0)$ alors
{
$c_{k-j} := g^{(k-j)}(x_0,\ldots,x_{k-j})$;
$M' := M' \cup \{M[j] + c_{k-j}X^{k-j}\}$;
}
}
$M'' := M'' \cup M'$;
$M' := [.]$;
}
$M := M''$;
$M'' := [.]$;
}
retourner M;

FIG. 5.7 – *Reconstructeur basique univarié*

Soit $\alpha \in \mathbb{F}_2^{64}$ un élément primitif. Pour obtenir une fonction satisfaisant cette propriété statistique, on a construit les applications naturelles :

$$h_0 : \mathbb{F}_{2^{64}} \longrightarrow \mathbb{F}_{2^{64}},$$

$$h_0(c_0 + c_1\alpha + \ldots + c_{63}\alpha^{63}) = c_0 + c_1\alpha + \ldots + c_{10}\alpha^{10},\ c_i \in \{0,1\}.$$

$$h_1 : \mathbb{F}_{2^{64}} \longrightarrow \mathbb{F}_2^{64},$$
$$h_1(c_0 + c_1\alpha + ... + c_{63}\alpha^{63}) = (c_0,c_1,...c_{63})$$

$$h_2 : \mathbb{F}_2^{64} \longrightarrow \mathbb{N},$$
$$h_2(c_0,c_1,...c_{63}) = c_0 + 2c_1 + 2^2c_2 + ... + c_{63}2^{63}.$$

L'image par $h_0 \circ h_1 \circ h_2$ des éléments $c_0 + 2c_1 + ... + c_{10}2^{10} \in \mathbb{F}_2^{64}$ est l'ensemble des entiers de 0 à $2^{11} - 1 = 2047$. On peut donc maintenant donner une description complète de cette fonction :

$$f(x) = \begin{cases} x^4 + x^2 + 1 & \text{si } h_0 \circ h_1 \circ h_2(x) \leq (1-\delta) \cdot 2^{11}, \\ x^{1000} & \text{si } h_0 \circ h_1 \circ h_2(x) > (1-\delta) \cdot 2^{11}. \end{cases}$$

Le testeur

On a donc maintenant utilisé le programme *Estime* (k,f,N) de la figure 5.4 avec dans notre cas $k = 4$. On a fait varier le nombre d'itérations N. On a donc représenté la quantité

$$\left(\mathbf{Prob}\left[\exists g \in \mathbb{F}_q^{(k)}[x] \text{ tel que } g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right]\right)^{\frac{1}{k+2}},$$

que l'on a mesurée par l'expérience en fonction du nombre d'itérations et de la distance $\delta = \Delta(f,RS[q-1,k+1])$ ainsi construite. Le degré k est fixé, donc le temps de calcul ne dépend que du nombre d'itérations N, on a donc donné ce temps de calcul en secondes.

$N/1-\delta$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	temps
50	0	0	0	0	0.521	0.584	0.702	0.835	0
10^2	0	0	0	0	0.521	0.669	0.728	0.813	0
5×10^2	0	0	0	0.398	0.521	0.625	0.727	0.780	0
10^3	0	0	0	0.413	0.547	0.589	0.706	0.813	0
5×10^3	0	0	0.290	0.421	0.486	0.605	0.697	0.798	0
10^4	0	0	0.290	0.426	0.486	0.603	0.703	0.798	1
5×10^4	0	0.164	0.283	0.397	0.509	0.598	0.698	0.800	5
10^5	0	0.164	0.294	0.396	0.503	0.597	0.701	0.800	11
5×10^5	0	0.179	0.304	0.399	0.499	0.600	0.700	0.800	54
10^6	0	0.200	0.300	0.400	0.500	0.600	0.700	0.800	110
5×10^6	0.108	0.200	0.300	0.400	0.500	0.600	0.700	0.800	550

FIG. 5.8 – *Estimation de la probabilité* $1 - \delta$

Dans cet exemple, on peut voir que l'on a

$$\textbf{Prob}\left[\exists g \in \mathbb{F}_q^{(k)}[x] \mid g(x_i) = f(x_i)\ \forall i \in \{0,\ldots,k+1\}\right] =$$

$$(1-\Delta(f,RS[q-1,k+1]))^{k+2}.$$

Cependant, comme l'avait prévu l'analyse théorique, cet algorithme n'est en aucun cas une amélioration par rapport à l'évaluation de la distance $\Delta(f,RS[q-1,k+1])$ que l'on aurait pu faire en ayant reconstruit le polynôme P grâce à l'algorithme de M. Sudan [45]. En fait cet algorithme reste exponentiel en le degré k.

Le reconstructeur

On va maintenant donner les résultats que l'on a obtenus avec le programme *Reconstructeur basique univarié*. On a repris la même fonction que précédemment. Rappelons que le programme prend en entrée les paramètres suivants: le degré k qui est égal à 4 dans notre cas, la fonction f, puis des paramètres qui vont déterminer le nombre d'itérations à effectuer dans cet algorithme. C'est ce paramètre N que l'on a fait varier pour obtenir le nombre minimum d'itérations nécessaire à la reconstruction du polynôme p. Chaque reconstruction a été répété 20 fois avec 20 succès. On a donné le temps de cette reconstruction en secondes.

$1-\delta$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
N	1000000	10000	1400	400	48	25	11	5	3
temps	899	12	1	0	0	0	0	0	0

La conclusion

On voit très clairement que les algorithmes que l'on a étudiés ne sont pas très performants du point de vue de la complexité, comparés à l'algorithme de Sudan [45], en fait ils sont exponentiels en le degré k. Il permettent seulement de donner le degré d'une fonction. Ces algorithmes mettent simplement en évidence la similitude entre le problème "Poly-agree" et le problème de reconstruction. Ces deux problèmes ont été introduits en début de chapitre. On va donc programmer l'algorithme de Sudan et l'appliquer sur le DES.

5.4.4 Application à la cryptanalyse du DES

On a rappelé la cryptanalyse de Jakobsen dans le chapitre 3 "Quelques attaques connues". Le DES ne possède pas une structure suffisamment simple pour reproduire directement l'attaque de Jakobsen. On va donc considérer la fonction de chiffrement du DES comme une boite noire, c'est-à-dire comme une fonction vue comme un polynôme univarié sur $\mathbb{F}_{2^{64}}$. Tout d'abord rappelons les

meilleurs algorithmes qui répondent au problème de reconstruction, c'est-à-dire l'algorithme Berlekamp-Welch et l'algorithme de Sudan [17].

Théorème 21. *[6] Soient n points $\{(x_i,y_i)\}_{i=1}^{n}$ tels qu'il existe un polynôme P de degré au plus k avec $y_i \neq P(x_i)$ pour au plus e valeurs de i, alors on a :*

1. *Il existe des polynômes $Q(x)$ et $Q'(x)$ où $\deg(Q) \leq k+e$ et $deg(Q') \leq e$ tels que pour tout $i \in \{1..n\}$, $Q(x_i) = y_i Q'(x_i)$.*
2. *Un tel couple de polynômes (Q,Q') peut être trouvé en $\mathcal{O}(n^2)$.*
3. *Pour tout couple $(Q,Q,)$, $\frac{Q}{Q'}(.) = P(.)$ si $e \leq \frac{1}{2}(n-k+1)$, (c'est-à-dire si $\Delta(f,RS[n,k+1]) < \frac{n-k+1}{2n}$).*

Remarque 11. *Le théorème précédent donne directement la distance*

$$\Delta(f,RS[q-1,k+1]) = \frac{\deg Q'}{n},$$

mais il n'est applicable que pour des petites distances :

$$\Delta(f,RS[n,k+1]) < \frac{n-k+1}{2n}.$$

Théorème 22. *[45] Soit $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$. Soient n le nombre de couples $(x_i,f(x_i))$ qui sont donnés. Soit k le degré des polynômes P que l'on veut reconstruire. Soit e le nombre maximal d'erreurs, c'est-à-dire le nombre maximal de couples $\{(x_i,f(x_i))\}$ pour lesquels $f(x_i) \neq P(x_i)$. Alors si*

$$e < n - \sqrt{2kn},$$

on peut reconstruire les polynômes P qui vérifient $|\{i \mid f(x_i) = P(x_i)\}| > n-e$. La complexité de l'algorithme permettant cette reconstruction est en $\mathcal{O}(n^2 \log^2(n))$ opérations arithmétiques sur le corps $\mathbb{F}_q$.

Remarque 12. *[30]Par simple application de l'inégalité de Chebyshev, on sait que pour utiliser le théorème précédent, il faut un échantillon de taille*

$$n = \frac{2k}{\delta^2}.$$

Lemme 12. *[45] Soient n points $\{(x_i,y_i)\}_{i=1}^{n}$ tels qu'il existe un polynôme bivarié Q satisfaisant :*

Le degré pondéré par rapport à $(1,k)$ de Q est au plus D, $Q \not\equiv 0$ et $\forall i \in \{1,\ldots,n\}$, $Q(x_i,y_i) = 0$, alors on a

1. *Un tel polynôme Q peut être déterminé en temps polynômial.*
2. *Si $P(x)$ est un polynôme de degré au plus k tel que $y_i \neq P(x_i)$ pour au plus $e < n-D$ valeurs de i alors le polynôme $(Y-P(X))$ divise $Q(X,Y)$.*

On va maintenant décrire un algorithme [20] qui permet de construire un tel polynôme $Q(x,y)$. Cet algorithme prend en entrée la distance δ, le degré k et la fonction f que l'on considère.

algo-reconstruct := fonction(δ,k,f)

$n := \frac{2k}{\delta^2}$;
$e := \delta n$;
$l := \lfloor \frac{n-e}{k-1} \rfloor$;

$G := \{1,y,y^2,\ldots,y^l\}$;(liste de polynômes)
$L := \{(x_i,f(x_i)),\ | \ i \in [1\ldots n]\}$;(ensemble de couples)

pour i de 1 à n faire
{
Soit $G[s]$,$s \in [0,\ldots,l]$ de degré minimal tel que $G[s](x_i,f(x_i)) \neq 0$

pour $t \in [0...l]\backslash s$ faire
{
$G[t] := G[t] - \frac{G[t](x_i,f(x_i))}{G[s](x_i,f(x_i))} \cdot G[s]$;
}

$G[s] := (x - x_i)G[s]$;
}
retourner G;

FIG. 5.9 - *Algorithme de reconstruction de $Q(x,y)$*

Par construction, cet algorithme retourne une liste de $l+1$ polynômes répondant aux critères du lemme 6. En prenant le polynôme $G[0]$ de la liste retournée par le programme 5.9, puis en le factorisant on reconstruit les polynômes de degré k. Dans notre cas, on a considéré la fonction itérée du DES: $F_K : \mathbb{F}_{2^{64}} \rightarrow \mathbb{F}_{2^{64}}$ avec une clef K fixée. On a cherché une approximation par un polynôme de degré 500, c'est-à-dire $k = 500$ par rapport aux notations utilisées jusqu'à présent, ce qui correspond à la dimension du code de Reed-Solomon. On a fait l'hypothèse que

$$\Delta(F_K,RS[2^{64}-1,k+1]) < \frac{20}{100}.$$

Donc avec ces paramètres, d'après ce que l'on a dit précédemment, $n = 5000$ couples de clairs-chiffrés sont nécessaires. Le nombre d'opérations nécessaires est de l'ordre de $1813564496 \sim 2^{30}$ opérations arithmétiques dans le corps $\mathbb{F}_{2^{64}}$. Le

coût d'une multiplication est de l'ordre de 64^2 opérations binaires. Donc le coût total est de l'ordre de 2^{43} opérations binaires. Avec un processeur P4 à 2 GHZ, le calcul a pris environ 5 heures. Malheureusement il n'y a pas de polynômes p de degré au plus 500 tel que $\Delta(F_K,p) < \frac{20}{100}$ pour la clef K tirée.

La conclusion

Il semble que cette approche ne soit pas efficace pour le DES. On a fait des simulations pour des degrés plus petits en faisant varier la clef, mais le résultat était le même, on n'a pas trouvé de polynômes univariés de bas degré approximant la fonction itérée du DES. Donc, il faudrait une avancée algorithmique importante pour espérer déceler une approximation par un polynôme de bas degré. Une autre possibilité est d'étudier les représentations du corps fini sur lequel on travaille, il est possible qu'il existe une représentation suivant laquelle, les polynômes à reconstruire soient de petits degré. Nous essaierons dans les prochains chapitres de répondre aux différents problèmes que nous nous sommes posé dans le contexte multivarié. On espère ainsi trouver des outils algorithmiques plus efficaces. Toutefois nous allons étudier la possibilité, plutôt que de considérer des entrées aléatoires, de prendre des entrées fortement corrélées, c'est à dire des entrées qui possèdent une forme particulière.

Chapitre 6

Test et reconstruction multivariés

Les problèmes considérés dans ce chapitre sont semblables aux problèmes du chapitre précédent. Nous étudierons simplement ici des fonctions du type $f : \mathbb{F}_q^m \rightarrow \mathbb{F}_q$.

6.1 Les problèmes considérés

On considère les fonctions du type $f : \mathbb{F}_q^m \rightarrow \mathbb{F}_q$ ($q = 2^t$) comme des boites noires pour lesquelles on ne connaît qu'une fraction des entrées-sorties. On peut considérer des entrées de deux types :

- les entrées de type clairs-connus définies dans le chapitre 1, c'est-à-dire que l'on suppose qu'une fraction aléatoire d'entrées-sorties nous est donnée,
- les entrées de type clairs-choisis définies dans le chapitre 1, c'est-à-dire que l'on choisit le type d'entrée que l'on désire : par exemple des éléments de $\mathbb{F}_q^m$ de la forme $(\overline{r},\overline{s}) \in \mathbb{F}_q^i \times \mathbb{F}_q^{m-i}$ avec les éléments $\overline{r}$ qui appartiennent à un sous-ensemble de $\mathbb{F}_q^i$.

On s'intéressera alors à deux types de problèmes :

- Le problème de reconstruction, défini dans le chapitre 1. En particulier, on aimerait construire des algorithmes répondant à ce problème, c'est-à-dire des algorithmes prenant en entrée les entiers r,T,l et un ensemble de couples pris aléatoirement :

$$P = \{(x_1,y_1),\ldots,(x_l,y_l)\}$$

tels que pour tout $i \in [1,\ldots,l]$ $x_i \in \mathbb{F}_q^m$, $y_i \in \mathbb{F}_q$. Cet algorithme doit ensuite retourner la liste de tous les polynômes $p : \mathbb{F}_q^m \rightarrow \mathbb{F}_q$ de degré total au plus r pour lesquels $p(x_i) = y_i$ pour au moins T valeurs de i distinctes. Nous n'avons pas d'algorithmes répondant au problème général, lorsque les entrées sont de type clairs-connus, aussi on limitera notre étude aux algorithmes fonctionnant à clair-choisis. En particulier on étudiera un algorithme dû à Goldreich et Levin dans le contexte $q = 2$ [16].

- Le problème *"Poly-agree"* défini dans le chapitre précédent. En particulier un algorithme qui répond à ce problème prend en entrée les entiers k,T,l et un ensemble de couples pris aléatoirement :

$$P = \{(x_1,y_1),\ldots,(x_l,y_l)\}$$

tels que pour tout $i \in [1,\ldots,l]$ $x_i \in \mathbb{F}_q^m$, $y_i \in \mathbb{F}_q$. Cet algorithme doit ensuite répondre à la question : existe-t'il au moins un polynôme $P : \mathbb{F}_q^m \rightarrow \mathbb{F}_q$ de degré total au plus k pour lequel $p(x_i) = y_i$ pour au moins T valeurs de i distinctes. Ces algorithmes seront comme dans le chapitre précédent appelés testeurs. Lorsque $\mathbb{F}_q$ est un corps premier, M. Sudan a développé dans sa thèse [44] de tels testeurs. Nous étendrons le domaine d'application de ces tests aux corps $\mathbb{F}_q$, avec $q = 2^t$.

Encore une fois nous n'avons pas d'algorithme pour le problème *"Poly-agree"* en général s'il n'est pas précédé d'une reconstruction. Donc, on s'intéressera essentiellement au problème de reconstruction dans le contexte clair-choisis; on verra cependant que les deux problèmes sont très liés. On a développé dans le chapitre 1, les notions de canal binaire symétrique et de canal adversaire; nous verrons que suivant le type de canal que l'on considère, le problème peut être plus ou moins difficile.

6.1.1 Les motivations

Parmi les applications possibles, il y a la recherche d'approximations affines de certaines fonctions coordonnées que constituent plusieurs tours de chiffrement du type DES. De tels outils pourraient par exemple être très intéressants pour les cryptanalyses linéaires. Ces concepts ont été introduit dans le chapitre 3.

6.1.2 Quelques notations

Nous utiliserons ici les mêmes notations que dans le chapitre 1. On considérera le corps $\mathbb{F}_q$ avec $q = 2^t$. L'ensemble des polynômes sur $\mathbb{F}_q$ en m variables de degré total au plus r sera noté

$$\mathbb{F}_q^{(r)}[X_1,\ldots,X_m].$$

Le code de Reed-Muller d'ordre r de longueur $n = q^m$ sur l'alphabet $\mathbb{F}_q$ sera noté

$$RM_q[r,m].$$

On a défini la distance de Hamming et la distance de Hamming normalisée entre deux mots de codes. On a aussi vu que l'image d'une fonction $f : \mathbb{F}_q^m \rightarrow \mathbb{F}_q$ constituait un mot de code, donc par abus de notation on notera

$$d(f,RM_q[r,m]) = \min\{d(\mathrm{Im}(f),c) \mid c \in RM_q[r,m]\},$$

la plus petite distance de Hamming qui existe entre un mot du code de Reed-Muller d'ordre r et l'image de f. On notera

$$\Delta(f,RM_q[r,m]) = \frac{d(f,RM_q[r,m])}{q^m},$$

la plus petite distance de Hamming normalisée qui existe entre un mot du code de Reed-Muller d'ordre r et l'image de f. Soit $g : \mathbb{F}_q^m \to \mathbb{F}_q$. On note encore par abus de notation :

$$d(f,g),$$

la distance de Hamming des mots de $\mathbb{F}_q^m$ induite respectivement par les images de f et de g. On notera de même la distance de Hamming normalisée entre f et g :

$$\Delta(f,g) = \frac{d(f,g)}{q^m}.$$

Rappelons que ces distances peuvent être interprétées comme des probabilités, on peut donc aussi écrire :

$$\Delta(f,g) = \underset{x\in\mathbb{F}_q^m}{\mathbf{Prob}}\left[f(x) = g(x)\right].$$

On étudiera surtout le cas $q = 2$, et lorsqu'il n'y aura pas d'ambiguité on notera $RM[r,m]$ le Reed-Muller d'ordre r et de longueur $n = 2^m$ sur le corps $\mathbb{F}_2$.

6.2 Testeurs multivariés

Dans cette section, on reprend exactement les mêmes notations que dans le chapitre précédent. On va reproduire le même type de test. On va tout d'abord rappeler les résultats de M. Sudan qui sont applicables sur un corps premier. On étendra ensuite ces résultats aux corps du type $\mathbb{F}_q$ avec $q = 2^t$.

6.2.1 Tests sur un corps premier

On considère, dans cette partie, les corps du type $\mathbb{F}_p$ avec p premier. On cherche ici à tester le degré d'une fonction $f : \mathbb{F}_p^m \to \mathbb{F}_p$. Il s'agit très simplement d'une généralisation des théorèmes vus dans le contexte univarié du chapitre précédent. En fait, il suffit simplement de réécrire les théorèmes dans le contexte multivarié, ce que nous allons faire. Ces résultats peuvent être trouvés dans la thèse de M. Sudan [44].

Définition 28. *On dit que les points* $\{x_0,\ldots,x_{r+1}\}$ *sont à espacements réguliers s'il existe* $h \in \mathbb{F}_p^m$ *tel que* $x_i = x_0 + i \cdot h$, *pour* $i \in \{0,\ldots,r+1\}$.

Rappelons que $\alpha_i = (-1)^{i+1}\binom{r+1}{i}$. Alors on a le lemme suivant :

Lemme 13. *Soient r,m des entiers positifs, et un nombre premier $p > mr$. Si $g : \mathbb{F}_p^m \rightarrow \mathbb{F}_p$ est un fonction telle que :*

$$\forall x,h \in \mathbb{F}_p^m \quad \sum_{i=0}^{r+1} \alpha_i g(x + i \cdot h) = 0,$$

alors g est polynôme en m variables de degré total au plus r.

Preuve : On va d'abord montrer que g est un polynôme de degré au plus r en chaque variable. Considérons des valeurs de h de la forme

$$h = (0,..,0,1,0,\ldots,0),$$

où la i-ème coordonnée est égale à 1. Alors, d'après le lemme 2 du chapitre 5, la fonction $g(v_1,\ldots,v_{i-1},y,v_{i+1},\ldots,v_m)$ est un polynôme de degré au plus r en y. Comme cela est vrai pour tout $i \in [1,\ldots,m]$ alors $g(x)$ est un polynôme multivarié de degré partiel au plus k en chaque variable. Maintenant on va montrer que le degré total est au plus k. Pour tout choix de $x,h \in \mathbb{F}_p^m$ le développement de chaque monôme de g donne lieu à un polynôme en la variable i, et comme par hypothèse $\sum_{i=0}^{r+1} \alpha_i g(x + i \cdot h) = 0$, le degré de g en i est au plus r. Donc le degré total de g est au plus r. □

On peut donc énoncer le théorème suivant :

Théorème 23. *Soient r,m des entiers positifs, un nombre premier $p > mr$ et une fonction $f : \mathbb{F}_p^m \rightarrow \mathbb{F}_p$ telle que*

$$\underset{x,h \in \mathbb{F}_p^m}{\mathbf{Prob}} \left[\sum_{i=0}^{r+1} \alpha_i \cdot f(x_i) = 0 \right] \geq 1 - \delta \quad où \quad \delta \leq \frac{1}{2(r+2)^2},$$

où la probabilité est prise sur des $r+2$ uplets à espacements réguliers. Alors $\Delta(f,RM_p[r,m]) \leq 2\delta$.

Preuve : La preuve est mot pour mot la même preuve que la preuve du théorème 16 du chapitre 5, il suffit de la réécrire dans le contexte multivarié. □

On en déduit maintenant le même genre de tests que dans le chapitre précédant, mais cette fois ci dans le cadre multivarié

Théorème 24. *Si les sorties d'une fonction $f : \mathbb{F}_p^m \rightarrow \mathbb{F}_p$ sont les images d'un polynôme de degré total au plus r alors la fonction passera le test à espacements réguliers. Si les images d'une fonction sont telles que $\Delta(f,RM_p[r,m]) > \frac{1}{(r+2)^2}$, alors avec probabilité $1 - \beta$, elle sera rejetée par le test à espacement régulier (3) 6.1.*

```
Test-régulier := fonction(r,β,f)

N := O(r^2 log(1/β));
réponse := oui;

    pour i de 1 à N
    {
      Prendre x,h ∈ F_{p^m} × F_{p^m}
      si Σ_{i=0}^{k+1} α_i · f(x + i · h) ≠ 0
      {
        réponse := non;
        break;
      }
    }
retourner réponse;
```

FIG. 6.1 – *Test à espacements réguliers (3)*

Preuve : Avec un probabilité de succès $1-\beta$ on peut tirer un couple $x,h \in \mathbb{F}_{p^m} \times \mathbb{F}_{p^m}$ tel que $\sum_{i=0}^{r+1} \alpha_i \cdot f(x+i\cdot h) \neq 0$ si $\underset{x,h\in\mathbb{F}_{p^m}\times\mathbb{F}_{p^m}}{\mathbf{Prob}} \left[\sum_{i=0}^{r+1} \alpha_i \cdot f(x_i) = 0\right] < 1 - \frac{1}{2(r+2)^2}$, c'est une conséquence directe du théorème 14 du chapitre 4. Si

$$\underset{x,h\in\mathbb{F}_{p^m}\times\mathbb{F}_{p^m}}{\mathbf{Prob}} \left[\sum_{i=0}^{r+1} \alpha_i \cdot f(x_i) = 0\right] < 1 - \frac{1}{2(r+2)^2},$$

alors le théorème 23 implique immédiatement que

$$\Delta(f,RM_p[r,m]) \leq \frac{1}{(r+2)^2}.$$

□

6.2.2 Extension aux corps binaires étendus

Les tests précédents ne sont malheureusement pas applicables directement pour les corps de type $\mathbb{F}_q$ avec $q = 2^t$ puisque la caractéristique du corps doit être supérieure au produit du nombre de variables par le degré que l'on considère. On se limitera à une étude en caractéristique 2, car c'est la caractéristique la plus courante dans le domaine des télécommunications et de la cryptographie à clef secrète. On va donc étendre ces résultats à la caractéristique 2. Il faut tout de

même noter que les résultats qui vont suivre ont un intérêt que si le corps que l'on considère est de cardinal plus grand que le degré total que l'on veut tester. On notera ω un élément primitif de $\mathbb{F}_{2^t}$ (ω doit engendrer le corps $\mathbb{F}_q$).

Définition 29. *On dira que les éléments* $\{x_0,\ldots,x_{r+1}\} \in \mathbb{F}_q^{r+2}$ *sont à espacements réguliers s'il existe* $x,h,\omega \in \mathbb{F}_q^m \times (\mathbb{F}_q^m)^* \times \mathbb{F}_q^*$, *tels que* $x_0 = x$ *et* $x_i = x + h \cdot \omega^{i-1}$ *pour* $i \in \{1,\ldots,r+1\}$. *(ω un élément primitif de $\mathbb{F}_{2^t}$)*

Lemme 14. *Soient r,m,t des entiers positifs tels que* $2^t > mr$. *Si* $g : \mathbb{F}_q^m \to \mathbb{F}_q$, $q = 2^t$ *est un fonction telle que :*

$$\forall x,h \in \mathbb{F}_q^m \times (\mathbb{F}_q^m)^* \quad \sum_{i=0}^{r+1} \alpha_i g(x + h \cdot \omega^{i-1}) = 0,$$

où les α_i *sont les éléments de* $\mathbb{F}_q$ *que l'on a construit dans le théorème 16 du chapitre précédent. Alors g est polynôme en m variables de degré total au plus r.*

Preuve : On va d'abord montrer que g est un polynôme de degré au plus r en chaque variable. Considérons des valeurs de h de la forme

$$h = (0,..,0,1,0,\ldots,0),$$

où la i-ème coordonnée est égale à 1. Alors, d'après le lemme 7 du chapitre 5, la fonction $g(v_1,\ldots,v_{i-1},y,v_{i+1},\ldots,v_m)$ est un polynôme de degré au plus k en y. Comme cela est vrai pour tout $i \in [1,\ldots,m]$, alors $g(x)$ est un polynôme multivarié de degré partiel au plus k en chaque variable. Maintenant, on va montrer que le degré total est au plus k. Pour tout choix de $x,h \in \mathbb{F}_q^m \times (\mathbb{F}_q^m)^*$ le développement de chaque monôme de g donne lieu à un polynôme en la variable ω, et comme par hypothèse $\sum_{i=0}^{r+1} \alpha_i g(x + h \cdot \omega^{i-1}) = 0$ le degré de g en ω est au plus r^2. Donc le degré total de g est au plus r. □

Le théorème précédent montre qu'il suffit d'effectuer des interpolations sur des $(r+2)$-uplets $(x_i,f(x_i))$, où les x_i sont à espacements réguliers pour vérifier qu'une fonction est de bas degré.

Théorème 25. *Soient r,m,t des entiers positifs tels que* $2^t > mr$, *et une fonction* $f : \mathbb{F}_q^m \to \mathbb{F}_q$, $q = 2^t$ *telle que*

$$\underset{x,h \in \mathbb{F}_q^m \times (\mathbb{F}_q^m)^*}{\mathbf{Prob}} \left[\sum_{i=0}^{r+1} \alpha_i \cdot f(x_i) = 0 \right] \geq 1 - \delta \quad \text{où} \quad \delta \leq \frac{1}{2(r+2)^2},$$

où la probabilité est prise sur des $(r+2)$*-uplets à espacements réguliers. Alors* $\Delta(f,RM_q[r,m]) \leq 2\delta$.

Preuve : C'est exactement la même preuve que la preuve du théorème 19. Il suffit simplement de remplacer les expression du type $i \cdot h$ par $h \cdot \omega^{i-1}$. □

Maintenant on peut déduire du théorème 25 un test de bas degré :

```
Test-régulier := fonction(r,β,f)

N := O(r² log(1/β));
réponse := oui;

    pour i de 1 à N
    {
      Prendre x,h ∈ F_{q^m} × F*_{q^m}
      si Σ_{i=0}^{k+1} α_i · f(x + i · h) ≠ 0
      {
        réponse := non;
        break;
      }
    }
retourner réponse;
```

FIG. 6.2 – *Test à espacements réguliers (4)*

Théorème 26. *Si les sorties d'une fonction f sont les images d'un polynôme de degré au plus r, alors la fonction passera le test à espacements réguliers. Si les images d'une fonction sont telles que $\Delta(f,RM_q[r,m]) > \frac{1}{(r+2)^2}$, alors avec probabilité $1-\beta$, la fonction sera rejetée par le test à espacements réguliers (2).*

Preuve : Avec un probabilité de succès $1-\beta$, on peut tirer un couple $x,h \in \mathbb{F}_q^m \times (\mathbb{F}_q^m)^*$ tel que $\sum_{i=0}^{r+1} \alpha_i \cdot f(x+i\cdot h) \neq 0$ si $\underset{x,h\in\mathbb{F}_q^m\times(\mathbb{F}_q^m)^*}{\mathbf{Prob}} \left[\sum_{i=0}^{r+1} \alpha_i \cdot f(x_i) = 0\right] < 1 - \frac{1}{2(r+2)^2}$, c'est une conséquence directe du théorème 14 du chapitre 4 "Quelques rappels de probabilité". Si

$$\underset{x,h\in\mathbb{F}_q^m\times(\mathbb{F}_q^m)^*}{\mathbf{Prob}} \left[\sum_{i=0}^{r+1} \alpha_i \cdot f(x_i) = 0\right] < 1 - \frac{1}{2(r+2)^2},$$

alors le théorème 14 implique immédiatement que

$$\Delta(f,RM_q[r,m]) \leq \frac{1}{(r+2)^2}.$$

□

La conclusion

Ces tests sont très pratiques pour tester le degré d'une fonction. Ils révèlent que de tester le degré d'une fonction sur un échantillon fortement corrélé, c'est-à-

dire des éléments ayant une forme bien particulière, suffit à caractériser le degré de cette fonction sur l'ensemble de son domaine de définition. Cependant ces tests restent faibles pour les problèmes cryptographiques qui nous intéressent car les corrélations sont plus faibles, et surtout, par construction ces tests ne sont absolument pas généralisables lorsque le corps sur lequel on travaille est le corps premier $\mathbb{F}_2$. On va donc passer du test à la correction. On verra qu'il existe des algorithmes de décodages très efficace pour le cas qui nous intéresse.

6.3 Le décodage

On se restreint ici à l'étude des fonctions du type $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$. On ne cherchera pas ici à répondre directement au problème "Poly-agree". On essaiera plutôt de déterminer la liste des polynômes p_i multivariés de degré k en m variables qui satisfont $\Delta(f,p_i) \leq \delta = \frac{1}{2} - \epsilon$. I s'agit d'un décodage par liste. Il est clair qu'en terme d'information, la résolution de ce problème est plus importante que la résolution du problème "Poly-agree", cependant, en pratique, il n'est pas sûr que reconstruire un polynôme soit plus difficile que de tester si ce polynôme existe vraiment. On parlera souvent dans la suite d'algorithmes adaptatifs, ce sont des algorithmes qui modifient leurs paramètres en fonctions des données qu'ils vont recueillir au cours de leur processus. Dans notre cas l'algorithme va poser une "question" à la fonction que l'on étudie (ses évaluations en certains points), puis il va agir en fonction de la réponse. Inversement on parlera d'algorithmes non adaptatifs lorsque les données relatives à cette fonction sont fixées et considérées comme paramètres de l'algorithme. Dans cette section, on fera une étude plus détaillée de l'algorithme de Goldreich-Levin qu'elle ne figure dans [16]. On donnera les différentes complexités de cet algorithme suivant le type de canal dans lequel on travaille. On proposera alors une variante de cet algorithme qui peut être plus efficaces selon les paramètres que l'on utilise. En particulier on donnera les résultats de nos simulations.

6.3.1 Rappel sur le décodage des Reed-Muller d'ordre 1

Nous allons tout d'abord rappeler des résultats bien connus sur le décodage des Reed-Muller d'ordre 1. Ce type de décodage s'applique aux deux situations auxquelles on a fait allusion : le canal binaire symétrique avec un bruit aléatoire et le bruit choisi par un adversaire. On considère donc une fonction $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$ en m variables $x_1, \ldots, x_m$. Les rappels que l'on va faire peuvent être trouvés dans des ouvrages classiques tels que [53]. Rappelons une des conséquences du théorème de Shannon qui sera très utile pour évaluer les performances des algorithmes que nous allons étudier.

Lemme 15. *(Shannon) [4]. Pour tout $\epsilon > 0$, si un algorithme de décodage pour un code C [n,m] sur $\mathbb{F}_2$ de longueur n et de dimension m peut décoder les mots p*

tels que $d(p,C) \leq (\frac{1}{2} - \epsilon)n$ *alors la complexité est au moins égale à* $min(n,m/\epsilon^2)$ *quand* $n \to \infty$.

On appelle matrice de Hadamard, la matrice carrée dont les coefficients sont les $(-1)^{u \cdot v}$, pour $u,v \in \mathbb{F}_2^m$. Pour simplifier, on notera $F(u)$ la fonction qui est égale à -1 si $f(u) = 1$ et 1 si $f(u) = 0$. On a donc

$$F(u) = (-1)^{f(u)},$$

la transformée d'Hadamard de F est donnée par

$$\hat{F}(u) = \sum_{v \in \mathbb{F}_2^m} (-1)^{u \cdot v} F(v) = \sum_{v \in \mathbb{F}_2^m} (-1)^{u \cdot v + f(v)}, \ u \in \mathbb{F}_2^m,$$

si $\hat{F}$ et F désignent les vecteurs lignes issus de l'évaluation sur le support $\mathbb{F}_2^m$, alors on a de manière équivalente

$$\hat{F} = FH,$$

et donc par conséquent on a

$$F = \frac{1}{2^m} \hat{F} H \ \text{ ou bien } \ F(v) = \frac{1}{2^m} \sum_{u \in \mathbb{F}_2^m} (-1)^{u \cdot v} \hat{F}(u)$$

on en déduit donc que

$$\hat{F}(u) = 2^m - 2d\left(f, \sum_{i=1}^{m} u_i x_i\right),$$

ou bien

$$d\left(f, \sum_{i=1}^{m} u_i x_i\right) = \frac{1}{2}\left(2^m - \hat{F}(u)\right), \ d\left(f, 1 + \sum_{i=1}^{m} u_i x_i\right) = \frac{1}{2}\left(2^m + \hat{F}(u)\right)$$

donc, la fonction affine $C(x_1, \ldots, x_m) = c_1 x_1 + \ldots c_m x_m$ qui correspond au mot de code du Reed-Muller d'ordre 1 le plus proche de f est telle que $\left|\hat{F}(C)\right|$ est maximum. Supposons maintenant, que la composante la plus grande est atteinte par $\left|\hat{F}(c_1, \ldots, c_m)\right|$, alors si $\hat{F}(c_1, \ldots, c_m) \geq 0$ on décode f par

$$\sum_{i=1}^{m} c_i x_i,$$

et inversement si $\hat{F}(c_1, \ldots, c_m) \leq 0$ on décode f par

$$1 + \sum_{i=1}^{m} c_i x_i,$$

Le produit direct $\hat{F} = FH$ coûte 2^{2m} additions; heureusement on peut utiliser la transformée de Fourier rapide qui coûtera $\mathcal{O}(n \log(n))$ additions dans $\mathbb{F}_2$ avec $n = 2^m$. Il reste ensuite a trier les $\hat{F}(u)$ ce qui coûte $\mathcal{O}(n \log(n))$ opérations avec un tri rapide.

La transformée de Fourier rapide discrète

Reprenons les notations précédentes, pour $u \in \mathbb{F}_2^m$:

$$\hat{F}(u) = \sum_{v \in \mathbb{F}_2^m} (-1)^{u \cdot v} F(v) = \sum_{v \in \mathbb{F}_2^m} (-1)^{u \cdot v + f(v)} = \sum_{v=0}^{2^m-1} F(v)(-1)^{u \cdot v},$$

et on peut écrire pour $u \in [1 \dots 2^{m-1}]$

$$\hat{F}(u) = \sum_{v=0}^{2^{m-1}-1} F(2v)(-1)^{2u \cdot v} + \sum_{v=0}^{2^{m-1}-1} F(2v+1)(-1)^{u \cdot (2v+1)}$$

et

$$\hat{F}(2^{m-1}+u) = \sum_{v=0}^{2^{m-1}-1} F(2v)(-1)^{2v \cdot (u+2^{m-1})} + \sum_{v=0}^{2^{m-1}-1} F(2v+1)(-1)^{(2v+1) \cdot (u+2^{m-1})}.$$

On veut donc calculer l'ensemble des $\hat{F}(u)$ pour $u \in [0, \dots, 2^m]$. L'algorithme que l'on va décrire est récursif. Maintenant, notons $X_x^n(u)$, la transformée de Fourier associée à la fonction $f(x)$ et de paramètre $n = 2^m$, $X_{2x}^n(u)$ désignera la transformée de Fourier associée à la fonction $f(2x)$ et de paramètre $n = 2^m$, alors les formules précédentes impliquent :

$$X_x^n(u) = X_{2x}^{n/2}(u) + (-1)^u X_{2x+1}^{n/2}(u),$$

$$X_x^n(\frac{n}{2} + u) = X_{2x}^{n/2}(u) - (-1)^u X_{2x+1}^{n/2}(u).$$

Donc les $X_x^n(u)$ peuvent être calculés au moyen des $X_{2x}^{n/2}(u)$ et $X_{2x+1}^{n/2}(u)$. On va donc décrire à la figure 6.3 l'algorithme récursif (algorithme de Cooley-Turkey [12]) qui permet de calculer ces $X_x^N(u)$[1]. L'initialisation de cette récursivité se fait à dim $= n$, on initialise alors un tableau a par

$$a := [(-1)^{f(0)}, (-1)^{f(1)}, \dots, (-1)^{f(n)}]).$$

Le résultat sera donner par a après le passage par la récursivité.

6.3.2 L'algorithme de Goldreich et Levin et son analyse

On va, dans cette partie, résumer les résultats de Goldreich et Levin [13] en caractéristique 2, qui ont été étendus par M. Sudan, Rubinfield et Goldreich [16] aux autres caractéristiques. Bien que très rapide, la transformée de Fourier reste impraticable pour les paramètres qui nous intéressent, par exemple $m = 64$ pour

1. http://cermics.enpc.fr/polys/info1/main/node97.html

```
FFT(dim ,a := [(−1)^f(0),(−1)^f(1),...,(−1)^f(n)]) (n = 2^m)

temporaire := 0;

si dim := 1 alors
  {
    retourner;
  }

dim := dim /2;
FFT(dim ,a([1... dim]));
FFT(dim ,a([dim +1...N]));

pour i de 1 à dim faire
  {
     temporaire := a[i];
     a[i] := temporaire + a[i + dim];
     a[i] := temporaire − a[i + dim];
  }

retourner;
```

FIG. 6.3 – *Transformée de Fourier discrète rapide*

les fonctions coordonnées du DES. En fait, la transformée de Fourier rapide reste de complexité exponentielle en m. On considère toujours la fonction $f : \mathbb{F}_2^m \to \mathbb{F}_2$, telle que $\Delta(f,RM[1,m]) \leq \frac{1}{2} - \epsilon$. On va donc décrire un algorithme polynômial en m et en ϵ pour le décodage des Reed-Muller d'ordre 1 et de longueur 2^m. Tout d'abord, on va définir une nouvelle métrique M. Cette métrique aura la propriété d'être invariante par la translation $+1$, c'est-à-dire que si f et g sont deux fonctions booléennes alors :

$$M(f,g) = M(f+1,g) = M(f,g+1) = M(f+1,g+1).$$

Si $g : \mathbb{F}_2^m \to \mathbb{F}_2$ désigne une fonction, alors :

$$M(f,g) = \min(\Delta(f,g),1 - \Delta(f,g)),$$

de même on définit la distance au code de Reed-Muller pour cette métrique :

$$M(f,RM[k,m]) = \min(\Delta(f,RM[k,m]),1 - \Delta(f,RM[k,m])).$$

Le principe de l'algorithme que nous allons étudier repose sur les propriétés suivantes.

Lemme 16. *Soit* $C(x_1,\ldots,x_m) = c_0+c_1x_1+\ldots+c_mx_m$, *une solution au problème posé, c'est à dire* $\Delta(f,C) \leq \frac{1}{2} - \epsilon$. *Soient* $\overline{s_j} = (s_j^{(i+1)},\ldots,s_j^{(m)}) \in \mathbb{F}_2^{m-i}$, $j \in [0,\ldots,2^{m-i}-1]$, *tous les éléments distincts de* $\mathbb{F}_2^{m-i}$. *Alors, pour tout* $1 \leq i \leq m$, *on a:*

$$\sum_{j=0}^{2^{m-i}-1} M(f(x_1,\ldots,x_i,\overline{s_j}),c_1x_1+\ldots+c_ix_i) \leq 2^{m-i}\left(\frac{1}{2}-\epsilon\right)$$

Preuve : Par hypothèse on a $\Delta(f,RM[1,m]) \leq \frac{1}{2}-\epsilon$ et $C(x_1,\ldots,x_m) = c_0+c_1x_1+\ldots+c_mx_m$ est une solution au problème. Il est équivalent de dire que

$$\underset{\overline{r},\overline{s}}{\mathbf{Prob}}\left[f(\overline{r},\overline{s}) = c_0+\sum_{j=1}^{i}c_jr_j+\sum_{j=i+1}^{m}c_js_{j-i}\right] \geq \frac{1}{2}+\epsilon,$$

où la probabilité est prise sur les couples $(\overline{r},\overline{s}) \in \mathbb{F}_2^i \times \mathbb{F}_2^{m-i}$, avec $\overline{r_j} = (r_j^{(1)},\ldots,r_j^{(i)})$. D'autre part, on a:

$$\underset{\overline{r},\overline{s}}{\mathbf{Prob}}\left[f(\overline{r},\overline{s}) = c_0+\sum_{j=1}^{i}c_jr_j+\sum_{j=i+1}^{m}c_js_{j-i}\right] =$$

$$1-2^{-m}\sum_{t=0}^{2^i-1}\sum_{l=0}^{2^{m-i}-1}\left[f(\overline{r_t},\overline{s_l})-\sum_{j=1}^{i}c_jr_j^{(t)}-\sum_{j=i+1}^{m}c_js_{j-i}^{(l)}-c_0\right] =$$

$$1-2^{-m}\sum_{l=0}^{2^{m-i}-1}\sum_{t=0}^{2^i-1}\left[f(\overline{r_t},\overline{s_l})-\sum_{j=1}^{i}c_jr_j^{(t)}-\sum_{j=i+1}^{m}c_js_{j-i}^{(l)}-c_0\right].$$

Maintenant, notons $\sigma_i(l) = \sum_{j=i+1}^{m}c_js_{j-i}^{(l)} - c_0 \in \mathbb{F}_2$, alors on a:

$$\underset{\overline{r},\overline{s}}{\mathbf{Prob}}\left[f(\overline{r},\overline{s}) = c_0+\sum_{j=1}^{i}c_jr_j+\sum_{j=i+1}^{m}c_js_{j-i}\right] =$$

$$1-2^{-m}\sum_{l=0}^{2^{m-i}-1}\sum_{t=0}^{2^i-1}\left[f(\overline{r_t},\overline{s_l})-\sum_{j=1}^{i}c_jr_j^{(t)}-\sigma_i(l)\right] \leq 1-2^{-m}.$$

$$\sum_{l=0}^{2^{m-i}-1}\min\left(\sum_{t=0}^{2^i-1}\left[f(\overline{r_t},\overline{s_l})-\sum_{j=1}^{i}c_jr_j^{(t)}\right],2^i-\sum_{t=0}^{2^i-1}\left[f(\overline{r_t},\overline{s_l})-\sum_{j=1}^{i}c_jr_j^{(t)}\right]\right).$$

On en déduit donc que:

$$\underset{\overline{r},\overline{s}}{\mathbf{Prob}}\left[f(\overline{r},\overline{s}) = c_0+\sum_{j=1}^{i}c_jr_j+\sum_{j=i+1}^{m}c_js_{j-i}\right] \leq$$

$$1 - 2^{-m+i} \sum_{l=0}^{2^{m-i}-1} M(f(x_1,\ldots,x_i,\overline{s_l}),c_1x_1 + \ldots + c_ix_i),$$

d'où le lemme. □

Par la suite, pour une fonction affine $C_i(x_1,\ldots,x_i) = c_0 + c_1x_1 + \ldots + x_i$, on posera :

$$\mathcal{E}^{(f)}(C_i) = 1 - 2^{-m+i} \sum_{l=0}^{2^{m-i}-1} M(f(x_1,\ldots,x_i,\overline{s_l}),C(x_1,\ldots,x_i)). \qquad (6.1)$$

Cette quantité n'est autre que un moins la valeur moyenne des distances normalisées

$$M(f(x_1,\ldots,x_i,\overline{s_l}),C(x_1,\ldots,x_i)).$$

Cette quantité nous servira à justifier la complexité des différents algorithmes que nous allons étudier.

Lemme 17. *([16]) Soit $C(x_1,\ldots,x_m) = c_0 + c_1x_1 + \ldots + c_mx_m$ une solution au problème posé, c'est-à-dire $\Delta(f,C) \leq \frac{1}{2} - \epsilon$. Alors, pour tout $i \in [1,\ldots,m]$, il y a une fraction au moins égale à $\frac{\epsilon}{2}$ des éléments $\overline{s_j} \in \mathbb{F}_2^{m-i}$ qui vérifient*

$$M(f(x_1,\ldots,x_i,\overline{s_j}),c_1x_1 + \ldots + c_ix_i) \leq \frac{1}{2} - \frac{\epsilon}{2}$$

Preuve : Désignons par ϵ_j la quantité

$$\epsilon_j = \frac{1}{2} - M(f(x_1,\ldots,x_i,\overline{s_j}),c_1x_1 + \ldots + c_ix_i).$$

Alors, d'après le lemme 16, on a $\frac{\epsilon_1+\ldots+\epsilon_{2^{m-i}}}{2^{m-i}} \geq \epsilon$. Supposons qu'il y ait t élément ϵ_j, tels que, $\epsilon_j > \frac{\epsilon}{2}$, par exemple $\epsilon_1,\ldots,\epsilon_t$, alors pour k valeurs de ϵ_j

$$\frac{t}{k} \geq \frac{\epsilon}{2}\left(\frac{1}{\frac{\epsilon_1+\ldots+\epsilon_t}{t} - \frac{\epsilon}{2}}\right).$$

Comme par hypothèse $1 \geq \frac{\epsilon_1+\ldots+\epsilon_t}{t} > \frac{\epsilon}{2}$, on a $\frac{t}{k} \geq \frac{\epsilon}{2}$, et le lemme est démontré. □

L'algorithme de Goldreich et Levin fonctionne en m étapes. A chaque étape, on a une notion de test, on suggère certains candidats et on teste s'ils ont une chance de provenir de véritables solutions au problème. Maintenant, décrivons ce test. Supposons qu'à la i-ème étape on ait déterminé les $i-1$ premiers coefficients $c_1,\ldots,c_{i-1}$. On va donc suggérer un coefficient $c_i = 0$ ou 1. Le candidat c_i passera ce test si et seulement si on tire un $\overline{s} \in \mathbb{F}_2^{m-i}$ tel que

$$M(f(x_1,\ldots,x_i,\overline{s}),c_1x_1 + \ldots + c_ix_i) \leq \frac{1}{2} - \frac{2\epsilon}{3},$$

où le paramètre $2\epsilon/3$ sera justifié par la taille de l'échantillon sur laquelle on va évaluer cette quantité. Si $c_1x_1 + \ldots + c_ix_i$ est une fonction linéaire qui provient d'une solution au problème $c_0 + c_1x_1 + \ldots + c_mx_m$, alors d'après le lemme 17, il suffit de tirer $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ éléments $\overline{s} \in \mathbb{F}_2^{m-i}$ pour en trouver un de convenable puisqu'il y en a une proportion au moins égale à $\epsilon/2$. La figure 6.4 représente donc le programme test que l'on peut trouver dans [16].

```
Test-candidat(f,ε,m,(c_1,...,c_i))

t :=def poly(m/ε);
t' :=def poly(m/ε);

   pour v de 1 à t faire
    {
      Prendre aléatoirement s̄ = (s_{i+1},...,s_m) ∈ F_2^{m-i};
     u := 0;
      pour k de 1 à t' faire
       {
         Prendre r̄ = (r_1,...,r_i) ∈ F_2^i;
         u := u + [f(r̄,s̄) − Σ_{j=1}^{i} c_j r_j];
       }
      si (max(u,t' − u) ≥ t' × (1/2 + ε/3))
       {
         (c_1,...,c_i) est accepté;
         break;
       }
    }
si (toutes les itérations ont été effectuées sans acceptation)
rejeter (c_1,...,c_i);
```

FIG. 6.4 – *Testeur de fonctions linéaires*

Dans cet algorithme, les paramètre $t \stackrel{def}{=} \text{poly}(m/\epsilon)$ et $t' \stackrel{def}{=} \text{poly}(m/\epsilon)$ ne sont absolument pas précisés dans l'article [16], et ne donnent qu'une idée très approximative de la complexité réelle. Toutefois, le lemme 17 permet de dire que $t = \mathcal{O}\left(\frac{1}{\epsilon}\right)$. On sait, d'autre part, que ce problème de reconstruction peut avoir plusieurs solutions, donc en particulier les deux coefficients $c_i = 0{,}1$ peuvent passer le programme *"Test-candidat"* à la i-ème étape. Nous allons donner une justification aux paramètres t,t' après la description complète de l'algorithme. L'algorithme de la figure 6.5 est supposé retourner une liste de fonctions linéaires contenant les fonctions linéaires C vérifiant $M(f,C) \leq \frac{1}{2} - 2\epsilon/3$.

```
Prog-reconstruct-linear(f,ε,m)

   R = [0]; (c'est une liste de fonctions affines).
   L = []; (c'est une liste de fonctions affines).

 pour i de 1 à m faire
    {
      pour v de 1 à card(R) faire
        {
           L = L ∪ (R[v] + x_i);
        }

      R = {};

      pour u de 1 à card(L) faire
        {
           R = R ∪ Test-candidat(f,ε,m,L[u]);
        }

      L := R;
    }

 retourner R;
```

FIG. 6.5 – *Algorithme de Goldreich-Levin*

Le programme qui nous intéresse doit retourner la liste des fonctions affines C' telles que $\Delta(f,C') \leq \frac{1}{2} - 2\epsilon/3$, mais lorsque l'on a obtenu une fonction linéaire C qui satisfait $M(f,C) \leq \frac{1}{2} - 2\epsilon/3$, il ne reste plus qu'a déterminer le terme constant c_0 pour obtenir la fonction affine qui est solution de notre problème de reconstruction. Pour obtenir ce terme constant il suffit par échantillonage de calculer une valeur approchée de la quantité :

$$A = \underset{x \in \mathbb{F}_2^m}{\mathbf{Prob}} \left[f(x) = C(x) \right].$$

Comme tout ce qui concerne l'échantillonage, ce calcul s'effectue avec une probabilité de succès et d'échec. Si cette quantité est supérieure à $1/2$, alors le terme constant de notre fonction affine sera 0, et dans le cas contraire, ce sera 1. On va maintenant étudier cet algorithme dans deux cadres très différents, c'est-à-dire le canal bruité de type canal binaire symétrique et le canal bruité par un adversaire. On a donné la définition de ces deux canaux dans le premier chapitre.

Le canal binaire symétrique

On peut en effet traduire le problème de reconstruction que l'on étudie, par un problème de décodage. On a vu, dans le chapitre 1 que la fonction que l'on étudie est en fait un mot du Reed-Muller d'ordre un, dont certaines positions sont erronées. Cette proportion de positions erronées est, selon nos notations, inférieure ou égale à $\frac{1}{2} - \epsilon$. Dans le cas du canal binaire symétrique, on fait l'hypothèse que ces positions sont uniformément distribuées. On se pose alors la question suivante :

Combien doit-on tirer de positions pour avoir une proportion de positions erronées égale à $\frac{1}{2} - \epsilon \pm \frac{\epsilon}{c}$, où $\frac{\epsilon}{c}$ représente l'écart type de l'erreur que l'on s'autorise par rapport à la quantité $\frac{1}{2} - \epsilon$? La taille de l'échantillon dépendra alors de la probabilité de succès que l'on désire, ainsi que des paramètres ϵ et c.

On a déjà répondu à cette question dans le chapitre 3. On va utiliser par commodité l'inégalité de Hoeffding pour majorer la taille de l'échantillon. On a

$$\mathbf{Prob}\left[\left|\frac{X_1 + \ldots + X_N}{N} - \left(\frac{1}{2} + \epsilon\right) \geq \frac{\epsilon}{c}\right|\right] \leq 2^{1 - \frac{2N\epsilon^2}{c^2 \log(2)}} \tag{6.2}$$

Donc, si on effectue $N > \frac{(u+1)c^2 \log(2)}{2\epsilon^2}$ tirages aléatoires, alors avec une probabilité d'échec au plus égale à 2^{-u}, on obtiendra une proportion de bits erronés égale à $\frac{1}{2} - \epsilon + \theta$ avec $|\theta| \leq \frac{\epsilon}{c}$.

Par construction, on a vu que si la taille des échantillons considérés étaient suffisante, alors les solutions de notre problème étaient incluses dans la liste de fonctions affines retournées par l'algorithme de Goldreich et Levin, avec toutefois

une certaine probabilité d'échec. D'autre part, on aimerait savoir de quelle manière la liste de fonctions affines peut croître au cours des étapes cet algorithme. Voici un résultat valable pour n'importe quel canal qui va nous permettre de répondre à cette question :

Théorème 27. *La borne de Johnson [16]. Soit* $\epsilon > 0$ *et* $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$. *On suppose que* $p_1, \ldots, p_k : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ *sont des fonctions affines distinctes satisfaisant*

$$\Delta(f,p_i) \leq \frac{1}{2} - \epsilon$$

pour tout $i \in \{1,\ldots,k\}$. *Alors* $k \leq \frac{1}{4\epsilon^2}$.

Donc, comme l'algorithme de Goldreich et Levin accepte les fonctions affines C qui vérifient $\Delta(C,f) \leq \frac{1}{2} - \frac{2\epsilon}{3}$, il retournera au plus $\frac{9}{4\epsilon^2}$ fonctions affines, avec une certaine probabilité d'échec. Par exemple il est possible qu'à la i-ième étape de l'algorithme de Goldreich-Levin la quantité que l'on doit estimer $M(f,c_1x_1 + \ldots + c_ix_i)$ soit sous évaluée auquel cas la solution dont les premier coefficients sont $\{c_1,\ldots,c_i\}$ sera omise. Il peut aussi se produire le cas inverse : la fonction linéaire $c_1x_1 + \ldots + c_ix_i$ n'est pas une solution à notre problème de reconstruction mais l'estimation par échantillonnage de la quantité $M(f,c_1x_1+\ldots+c_ix_i)$ est surévaluée auquel cas l'algorithme de Goldreich-Levin retournera la fonction linéaire dont les premiers coefficients sont $c_1x_1+\ldots+c_ix_i$. Évidemment, plus l'écart-type de l'erreur que l'on s'autorise est grande (ce que l'on a représenté par ϵ/c dans la formule 6.2), plus la liste de candidats potentiels au cours des étapes de l'algorithme de Goldreich-Levin pourra croître. On verra que grâce aux résultat récents de Tor Helleseth, Torleiv Klove, et Vladimir I. Levenshtein [28], ces effets seront assez limités dans le canal binaire symétrique.

La probabilité d'obtenir une liste de solutions

On va résumer les travaux récents de Tor Helleseth, Torleiv Klove, et Vladimir I. Levenshtein [28]. Ces travaux permettent d'affirmer que la probabilité d'obtenir plus d'une solution C satisfaisant $\Delta(f,C) \leq \frac{1}{2} - \epsilon$ tend vers 0 lorsque le nombre de variables m tend vers l'infini.

Notons $B_C(t)$ l'ensemble des erreurs corrigibles de poids t pour un code linéaire C de paramètres $[n,k,d]$, c'est à dire l'ensemble des erreurs E pour lesquelles le seul mot a du code C considéré qui vérifie $\Delta(a,c+E) \leq t/n$ pour tout $c \in C$ fixé est c. Notons alors $R_C(t) = |B_C(t)|/\binom{n}{t}$, $t \in [0,\ldots,n]$ la fraction d'erreurs corrigibles parmi toutes les erreurs.

Théorème 28. *[28] Soit* $\{C_n\}$ *une suite de code* $[n,k,d]$ *tel que* $d/n \to \mu$ *quand* $n \to \infty$ *où* $0 < \mu \leq 1/2$. *Alors pour tout* $s = s(n)$ *tel que* $\sqrt{n} \leq s < n$,

$$1 - R_C\left(\left\lfloor \frac{n-s}{2} \right\rfloor\right) \lesssim \frac{1}{s}\sqrt{\frac{(1-\mu)n}{2\pi\mu}}2^k e^{-\frac{\mu s^2}{2(1-\mu)n}}.$$

Donc en particulier ce théorème est applicable aux codes de Reed-Muller. Replaçons nous dans notre contexte. On considère un code de Reed-Muller de paramètres $[n,k = 1 + m + \ldots + \binom{m}{r},d = 2^{m-r}]$. Le poids t de notre erreur est noté $n(1/2 - \epsilon)$. Donc $(n - s)/2 = t$ implique que $s = n\epsilon/2$. On a donc

$$1 - R_C\left(\lfloor n(1/2 - \epsilon)\rfloor\right) \lesssim \frac{1}{\epsilon}\sqrt{\frac{2n(2^r - 1)}{2^r\pi}}e^{-\frac{n\epsilon^2}{2^{r+1}}}. \tag{6.3}$$

En particulier, on est dans le cas où $r = 1$, on a donc dans ce cas l'inégalité

$$1 - R_C\left(\lfloor n(1/2 - \epsilon)\rfloor\right) \lesssim \frac{1}{\epsilon}\sqrt{\frac{n}{\pi}}e^{-\frac{n\epsilon^2}{4}}.$$

Tout ceci montre que si l'on considère un nombre suffisant de variables m, tel que $2^m > 4/\epsilon^2$ à l'issue de notre décodage, il ne restera qu'une seule solution à notre problème avec une forte probabilité.

Théorème 29. *Soient $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ tel que $\Delta(f,RM[1,m]) \leq \frac{1}{2} - \epsilon$. Supposons que la fonction f est équivalente à une fonction affine dont on aurait perturbé aléatoirement une fraction d'au plus $\frac{1}{2} - \epsilon$ de ses images. Supposons, de plus, que le nombre de variables soit si grand ($2^m \gg \epsilon^{-2}$), qu'avec une très grande probabilité, il n'y ait qu'une seule fonction affine C qui vérifie $\Delta(f,C) \leq 1/2 - \epsilon$. Alors, le problème de reconstruction à clairs choisis peut être résolu avec une complexité de l'ordre de*

$$\mathcal{O}\left((\log^2(m) + u^2)\left(\frac{\log^2(1/\epsilon^2)}{\epsilon^5} + \frac{(m - \log(1/\epsilon^2))^2}{\epsilon^3}\right)\right)$$

opérations binaires et avec une probabilité d'échec inférieure à 2^{-u}.

Preuve : On va premièrement fixer quelques notations. Soit $EV_i^t(C)$ l'évènement "un bon candidat C est rejeté à la ième étape au bout de t essais". On a vu 17 qu'il y avait au moins une fraction ϵ', des éléments $s \in \mathbb{F}_2^{m-i}$ tels que $M(f(x,s),C(x,s)) \leq 1/2-\epsilon'$, $x \in \mathbb{F}_2^i$. Donc la probabilité pour qu'un bon candidat C soit rejeté durant les m étapes est :

$$\mathbf{Prob}\left[EV_1 \cup \ldots \cup EV_m\right].$$

On a $\mathbf{Prob}\left[EV_i^t(C)\right] \leq (1 - \epsilon')^t \leq e^{-t\epsilon'}$. On peut maintenant en déduire :

$$\mathbf{Prob}\left[EV_1 \cup \ldots \cup EV_m\right] \leq me^{-t\epsilon'}.$$

On aimerait que cette quantité tende vers 0 très rapidement, donc on aimerait que $me^{-t\epsilon'} < 2^{-u'}$, ce qui implique : $t > (\log(m) + u'\log(2))/\epsilon'$. Maintenant reprenons la formule 6.3 et déterminons le nombre d'étapes nécessaires pour obtenir un décodage unique avec une probabilité supérieur à $1 - 2^{-v}$:

$$1 - R_C\left(\lfloor i(1/2 - \epsilon')\rfloor\right) \lesssim \frac{1}{\epsilon'}\sqrt{\frac{i}{\pi}}e^{-\frac{i\epsilon'^2}{4}} \leq 2^{-v},$$

ce qui implique assymptotiquement que

$$2\log(i) - 2\log(\pi) + 4v\log(2) - 4\log(\epsilon') \leq i\epsilon'^2,$$

donc si

$$i > \frac{-2\log(\pi) + 4v\log(2) - 4\log(\epsilon')}{\epsilon'^2},$$

alors $1 - R_C(\lfloor i(1/2 - \epsilon')\rfloor) \leq 2^{-v}$. Il faut maintenant déterminer la taille de l'échantillon nécéssaire pour évaluer $M(f,C)$. Soit $X_i = f(x,s_i) - C(x,s_i)$ la variable aléatoire qui est égale à 0 avec une probabilité $1/2 + \epsilon$. L'inégalité de Hoeffding nous dit que:

$$\textbf{Prob}\left[\left|\frac{X_1 + \ldots + X_N}{N} - \left(\frac{1}{2} + \epsilon\right) \geq \frac{\epsilon}{c}\right|\right] \leq 2^{1 - \frac{2N\epsilon^2}{c^2\log(2)}}.$$

On veut une probabilité d'échec qui décroît exponentiellement avec la taille de l'échantillon, soit:

$$2^{1 - \frac{2N\epsilon^2}{c^2\log(2)}} \leq 2^{-w}.$$

Donc si $N \geq c^2(\log(2) + w)/\epsilon^2$, cette condition est satisfaite. Cette inégalité nous dit deux choses, si $2^i \geq c^2(\log(2) + w)/\epsilon^2$, alors comme on est dans les conditions du canal binaire symétrique: $M(f(x,s_i),C(x,s_i)) \leq 1/2 - \epsilon \pm \epsilon/c$ avec probabilité supérieur à $1 - 2^{-w}$. D'autre part cette formule donne la taille d'échantillon nécessaire pour évaluer un distance la probabilité de succès que l'on souhaite. Comme il y a m étapes il faut évaluer la probabilité que l'évènement $FV(C)$ "une distance $M(f,C)$ est mal évaluée durant ces m étapes", soit:

$$m2^{1 - \frac{2N\epsilon^2}{c^2\log(2)}}.$$

Donc si $N > c^2(\log(2m) + w')/\epsilon^2$, la probabilité d'une distance mal évaluée durant les m étapes est inférieur à $2^{-w'}$. Donc finalement la probabilité que C soit rejeté au cours des m étapes est la probabilité de l'union des évènements $EV_1(C) \cup \ldots \cup EV_m(C) \cup FV(C)$. Donc si $t > (\log(m) + (\alpha + 1)\log(2))/(\epsilon - \epsilon/c)$ et si $N > c^2(\log(2m) + (\alpha + 1))/\epsilon^2$, alors le candidat C franchira les m étapes avec une probabilité de succès supérieur à $1 - 2^{-\alpha}$. Il reste à décrire les i premières étapes de l'algorithme. Pendant ces i premières étapes, la liste des candidats peut en effet croître pour atteindre une taille de l'ordre $\mathcal{O}(\epsilon^{-2})$. On a fixer correctement les paramètres t,t' du programme "Test-candidat", donc on va pouvoir maintenant donner la complexité de l'algorithme. Pour les i première étapes on obtient:

$$\mathcal{O}\left((\log^2(m) + u^2)\frac{\log^2(1/\epsilon^2)}{\epsilon^4}\right).$$

Puis pour les $m - i$ restantes, on a montré la liste des candidats potentiels se limitait à un unique candidat, donc en déduit la complexité:

$$\mathcal{O}\left((\log^2(m) + u^2)\frac{(m - \log(1/\epsilon^2))^2}{\epsilon^2}\right).$$

Donc le nombre d'opérations binaires à effectuer pour retourner la liste des fonctions affines est inférieur ou égal à

$$\mathcal{O}\left((\log^2(m)+u^2)\left(\frac{\log^2(1/\epsilon^2)}{\epsilon^5}+\frac{(m-\log(1/\epsilon^2))^2}{\epsilon^3}\right)\right)$$

pour une probabilité d'échec inférieure à 2^{-u}. □

Remarque 13. *Dans la pratique on se rend vite compte que si le nombre de variable est suffisant (≈ 20) et si ϵ n'est pas trop petit ($\epsilon > 0.01$), et toujours dans le contexte du canal binaire symétrique, il est rare d'avoir plusieurs solutions. Avec 1000 essais, pas une seule fois on a obtenu plus d'une solution. On présentera ultérieurement une étude expérimentale de cet algorithme.*

Le canal adversaire

Dans ce cas, notre seule hypothèse est $\Delta(f,RM[1,m]) \leq \frac{1}{2}-\epsilon$, mais la répartition des erreurs n'est pas spécialement uniforme. D'après le lemme 17, si la taille de l'échantillon est suffisant, à la i-ième étape de l'algorithme, un candidat $C_i(x_1,\ldots,x_i) = c_0 + \ldots + c_i x_i$ qui provient d'une véritable solution à notre problème passera le test *"Test-candidat(f,ϵ,m,C_i)"* de la figure 6.4. Donc, l'algorithme de Goldreich et Levin retournera bien la liste des fonctions linéaires C qui vérifient $M(f,C) \leq \frac{1}{2}-\frac{2\epsilon}{3}$. Il faut maintenant donner la complexité de cet algorithme pour ce type de canal, voici le théorème de Goldreich-Rubinfield-Sudan que l'on trouve dans [16]:

Théorème 30. *[16] Soient $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ et les paramètres ϵ,k. Alors l'algorithme de Goldreich et Levin retournera avec une complexité égale à poly($\frac{km}{\epsilon}$), et avec une probabilité de succès au moins égale à $1-2^{-k}$ une liste qui contiendra la liste des fonctions affines qui coïncident avec au moins une fraction $\frac{1}{2}+\epsilon$ des entrées, mais qui ne contiendra pas de fonctions affines coïncidant avec une fraction moindre que $\frac{1}{2}+\frac{\epsilon}{4}$ de ces entrées.*

L'argument qui est invoqué pour justifier une complexité polynomiale est donné par le théorème 27, c'est la borne de Johnson. Mais est-ce suffisant? On peut se poser la question légitime suivante : la liste de candidats ne pourait-elle pas être exponentielle ou sous-exponentielle au cours des étapes de l'algorithme. On va démontrer que la liste des candidats peut croître jusqu'au moins une certaine borne :

Théorème 31. *Donnons aux paramètres t,t' du programme 6.4 les valeurs respectives poly(m/ϵ) et $\mathcal{O}(\epsilon^{-2})$. La deuxième quantité est justifiée par l'inégalité de Chebyshev. On peut construire une fonction $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ telle que*

$$\Delta(f,RM[1,m]) \leq 1/2-\epsilon$$

et telle que, lors des étapes de l'algorithme de Goldreich et Levin, la liste des candidats atteigne une taille de l'ordre de

$$poly\left(\frac{m}{\epsilon}\right)/\epsilon^2.$$

Preuve : Supposons que la liste de candidats atteint sa taille maximale à la i-ième étape (donc inférieur à 2^i). Soit t fonctions booléennes $g_j : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$ telles qu'il existe, pour chacune, $1/4\epsilon^2$ fonctions linéaires distinctes contenant la solution nulle $C = 0$ et qui sont à égale distance de la fonction g_j, c'est-à-dire $\Delta = 1/2 - \epsilon$. On supposera, de plus, que toutes ces fonctions linéaires sont distinctes dans leur ensemble, exceptée la fonction nulle. On va donc construire une fonction booléenne h qui vérifie $\Delta(h,0) \leq \frac{1}{2} - \epsilon$ en construisant ses images. Les antécédents de cette fonction sont de la forme $(\overline{r},\overline{s}) \in \mathbb{F}_2^i \times \mathbb{F}_2^{m-i}$. On va partager les 2^{m-i} éléments $\overline{s}$ en t sous-ensembles de taille $(2^{m-i})/t$. On considère alors les t sous-ensembles d'éléments

$$\{(\overline{r},\overline{s}) \in \mathbb{F}_2^i \times \mathbb{F}_2^{m-i}, |\overline{r}| = 2^i,\ |\overline{s}| = (2^{m-i})/t\}$$

Sur chacun de ces t sous-ensembles on décide que notre fonction h est l'évaluation des fonctions g_j respectivement sur chacun de ces ensembles. Donc, par construction si $2^i > t/\epsilon^2$, la liste va croître au moins jusqu'à t/ϵ^2.

□

Remarque 14. *Avec les paramètres que l'on a choisi pour donner une complexité précise à l'algorithme de Goldreich-Levin [13], c'est-à-dire avec le paramètre $t = \mathcal{O}(\epsilon^{-1})$ de la figure 6.5, la liste peut atteindre une taille de l'ordre de $\mathcal{O}(\epsilon^{-3})$. Donc la complexité de cet algorithme dans le pire cas est d'au moins $\mathcal{O}(m^2\epsilon^{-6})$.*

On a pu construire une fonction booléenne pour laquelle la liste des candidats suggérés pendant les étapes de l'algorithme de Goldreich-Levin atteint une taille de l'ordre de $poly\left(\frac{m}{\epsilon}\right)/\epsilon^2$, on va maintenant montrer qu'il s'agit d'une borne.

Théorème 32. *Considérons l'algorithme de Goldreich-Levin de la figure 6.5 avec les paramètres $t = poly(m/\epsilon)$, et $t' = \mathcal{O}(\epsilon^{-2})$ fixés pour le testeur de fonctions linéaires de la figure 6.4. Considérons une fonction $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$ telle que*

$$\Delta(f,RM[1,m]) \leq 1/2 - \epsilon.$$

Alors la liste des candidats suggérés au cours de cet algorithme n'excédera pas une taille de $poly\left(\frac{m}{\epsilon}\right)/\epsilon^2$.

Preuve : Par construction, à l'étape i de l'algorithme de Goldreich-Levin, les candidats C_j suggérés seront acceptés avec une grande probabilité si et seulement s'il existe pour chacun, $\mathcal{O}(poly^{-1}(m\epsilon^{-1})2^{m-i})$ éléments $\overline{s}_j \in \mathbb{F}_2^{m-i}$ tels que

$$M(f(x_1,\ldots,x_i,\overline{s}_j),C_j) \leq 1/2 - 2\epsilon/3.$$

On sait de plus que pour un élément $\overline{s} \in \mathbb{F}_2^{m-i}$, il y a au plus $b = 9\epsilon^{-2}/16$ fonctions affines C telles que $\Delta(f(x_1,\ldots,x_i,\overline{s}),C) \leq 1/2 - 2\epsilon/3$. On va donc montrer par récurrence sur k que pour $t = poly(m\epsilon^{-1})$, et k ensembles de $t^{-1} \times 2^{m-i}$ éléments $\overline{s}$ de $\mathbb{F}_2^{m-i}$, il existe au plus $9k\epsilon^{-2}/16$ fonctions affines telles que pour chacune d'entre elles, il y ait $t^{-1} \times 2^{m-i}$ éléments $\overline{s}$ de $\mathbb{F}_2^{m-i}$ contenus dans les k ensembles, tels que

$$M(f(x_1,\ldots,x_i,\overline{s}),C_j) \leq 1/2 - 2\epsilon/3, \tag{6.4}$$

et pour cette borne $(9k\epsilon^{-2}/16)$ les ensembles sont disjoints, on ajoutera dans l'hypothèse de récurrence que le rapport

$$R_k = \frac{\text{nombre maximal de fonctions affines distinctes}}{\text{nombre total d'éléments contenus dans les } k \text{ ensembles}}$$

soit maximum lorsque les k ensembles sont disjoints, c'est-à-dire encore égale à

$$\frac{9t}{16\epsilon^2 2^{m-i}} = \frac{bt}{2^{m-i}}.$$

Pour $k = 2$, notons E_1 et E_2 les ensembles contenant $t^{-1} \times 2^{m-i}$ éléments de $\mathbb{F}_2^{m-i}$. Alors il y a deux possibilités : soit E_1 et E_2 sont disjoints, auquel cas la borne de Johnson nous donne la réponse, pour chacun des ensembles il y a au plus $9\epsilon^{-2}/16$ fonctions affines vérifiant la condition 6.4 et $R_k = \frac{bt}{2^{m-i}}$. Maintenant si les ensembles ne sont pas disjoints, alors dans ce cas on va montrer qu'il y a au plus $b = 9\epsilon^{-2}/16$ fonctions affines vérifiant la condition 6.4 sur les deux ensembles E_1 et E_2 réunis. Si l'on suppose qu'il y a $b = 9\epsilon^{-2}/16$ fonctions affines vérifiant la condition 6.4 sur E_1, alors comme $b = 9\epsilon^2/16$ est une borne, on en déduit que $E_1 = E_2$ (les ensembles sont confondus). Supposons qu'il y ait $b-\alpha > 0$ fonctions affines qui vérifient la condition 6.4 pour E_1, alors comme l'intersection de E_1 et E_2 sont non vides, il y a au plus α fonctions affines qui vérifient la condition 6.4 pour E_2, dans le cas contraire on aurait des éléments de $E_2 \cap E_1$ pour lesquelles on aurait plus de b fonctions affines qui vérifieraient la condition 6.4, ce qui est absurde. Donc si $E_2 \cap E_1$ est non vide il y a au plus b fonctions affines qui vérifient la condition 6.4 pour $E_1 \cup E_2$, il reste à montrer que le rapport $R_k < \frac{bt}{2^{m-i}}$ dans ce cas : comme les deux ensembles ne sont pas disjoints, il y a strictement moins de $2t^{-1} \times 2^{m-i}$ éléments dans $E_1 \cup E_2$ et strictement plus que $t^{-1} \times 2^{m-i}$ car les deux ensembles ne sont pas confondus, d'autre part on viens de montrer qu'il y avait au plus b fonction affines vérifiant la condition 6.4, donc $R_k < \frac{bt}{2^{m-i}}$. L'hypothèse est donc vérifiée pour $k = 2$. On suppose maintenant qu'elle est vraie jusqu'au rang k, et on va la démontrer au rang $k+1$. On suppose maintenant que l'on a $k+1$ ensembles $(E_s)_{s\in[1,k+1]}$. Soit $S_k = E_1 \cup E_2 \cup \ldots E_k$. Par hypothèse de récurrence, pour S_k il y a au plus $9k\epsilon^{-2}/16$ fonctions affines pour lesquelles il existe au moins $2^{m-i}(poly(m\epsilon^{-1}))^{-1}$ éléments $\overline{s}_j \in \mathbb{F}_2^{m-i}$ tels que

$$M(f(x_1,\ldots,x_i,\overline{s}_j),C_j) \leq 1/2 - 2\epsilon/3,$$

et on a exactement $9k\epsilon^{-2}/16$ fonctions affines si et seulement si les k ensembles $E_1,...,E_k$ sont disjoints. Plusieurs cas sont possibles :

1. Si S_k est formé d'ensembles disjoints, alors par hypothèse de récurrence pour S_k, $R_k = \frac{bt}{2^{m-i}}$ et il y a exactement $9k\epsilon^{-2}/16$ fonctions affines qui vérifient la condition 6.4. Il y a maintenant deux possibilités pour E_{k+1}.
   (a) Si S_k et E_{k+1} sont disjoints alors $R_{k+1} = \frac{bt}{2^{m-i}}$ et il y a donc au maximum $9(k+1)\epsilon^{-2}/16$ fonctions affines qui vérifient la condition 6.4.
   (b) Si S_k et E_{k+1} ne sont pas disjoints alors comme pour le cas $k = 2$ le nombre de fonction affines constructibles pour l'ensemble E_{k+1} est strictement inférieur à b, et comme E_{k+1} n'est pas inclus dans S_k, il y a strictement moins de $(k+1)t^{-1} \times 2^{m-i}$ éléments dans $S_{k+1} = S_k \cup E_k$ et strictement plus que $kt^{-1} \times 2^{m-i}$. Donc $R_{k+1} < \frac{bt}{2^{m-i}}$ et il y a strictement moins de $9(k+1)\epsilon^{-2}/16$ fonctions affines distinctes qui vérifient la condition 6.4.
2. Si S_k est quelconque, alors par hypothèse de récurrence pour S_k, $R_k < \frac{bt}{2^{m-i}}$. On a alors deux cas de figure :
   (a) Si S_k et E_{k+1} ne sont pas disjoints alors comme pour le cas $k = 2$ le nombre de fonctions affines constructibles pour l'ensemble E_{k+1} est strictement inférieur à b, et comme E_{k+1} n'est pas inclus dans S_k, il y a strictement moins de $(k+1)t^{-1} \times 2^{m-i}$ éléments dans $S_{k+1} = S_k \cup E_k$ et strictement plus que $kt^{-1} \times 2^{m-i}$. Donc $R_{k+1} < \frac{bt}{2^{m-i}}$ et il y a strictement moins de $9(k+1)\epsilon^{-2}/16$ fonctions affines distinctes qui vérifient la condition 6.4.
   (b) Si S_k et E_{k+1} sont disjoints, alors comme le nombre de fonctions affines constructibles pour l'ensemble S_k est strictement inférieur à $9k\epsilon^{-2}/16$, donc on en déduit que $R_{k+1} < \frac{bt}{2^{m-i}}$ et qu'il y a strictement moins de $9(k+1)\epsilon^{-2}/16$ fonctions affines distinctes qui vérifient la condition 6.4.

La propriété est démontré et implique qu'on aura un maximum de fonction affine en considérant des ensembles disjoints formés de $t^{-1} \times 2^{m-i}$ éléments $\overline{s}$ de $\mathbb{F}_2^{m-i}$. A la i-ième étape, on a au plus $t = poly(m\epsilon^{-1})$ ensembles disjoints formés de $t^{-1} \times 2^{m-i}$ éléments $\overline{s}$ de $\mathbb{F}_2^{m-i}$ vérifiant la condition 6.4, donc la liste des candidats n'excédera pas $poly(m/\epsilon)\epsilon^{-2}$ au cours des étapes de l'algorithme de Goldreich-Levin.

□

On peut donc maintenant donner le théorème final :

Théorème 33. *Soient $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ tel que $\Delta(f, RM[1,m]) \leq \frac{1}{2} - \epsilon$. Alors le problème de reconstruction à clairs choisis dans le canal adversaire peut être résolu avec une complexité de l'ordre de*

$$\mathcal{O}\left(\frac{m^2(\log^2(m/4\epsilon^2) + u^2)}{\epsilon^6}\right)$$

opérations binaires et avec une probabilité d'échec inférieure à 2^{-u} avec l'algorithme de Goldreich-Levin 6.5.

Preuve : La méthode est la même que dans la preuve du théorème 29, simplement ici il faut considérer une liste de taille la borne de Johnson, $1/4\epsilon^2$. Il faut donc substituer à m la valeur $m/4\epsilon^2$ car il s'agit de calculer dans ce cas là, la probabilité pour qu'un candidat parmi les $1/4\epsilon^2$ soit rejeté pendant les m étape de l'algorithme. Bien sur, il ne faut pas oublier que la taille de la liste peut atteindre $2/\epsilon^3$. On en déduit donc une complexité de l'ordre de

$$\mathcal{O}\left(\frac{m^2(\log^2(m/4\epsilon^2)+u^2)}{\epsilon^6}\right),$$

et avec une probabilité de succès supérieure ou égale à $1-2^{-u}$. □

La conclusion

Cet algorithme est très efficace, on peut cependant remarquer que la liste des candidats au cours de l'algorithme peut dépasser la borne de Johnson, ce qui augmente la complexité de l'algorithme, et augmente la mémoire nécessaire. En pratique cet algorithme semble relativement lent puisque une liste assez large subsiste pendant la majeure partie des m étapes de l'algorithme. On va donc introduire une nouvelle quantité pour modifier cet algorithme en espérant que la taille de la liste diminue plus rapidement.

6.3.3 L'algorithme de Goldreich et Levin revisité

On va donner quelques modifications à l'algorithme de Goldreich et Levin afin d'améliorer sa complexité. Une première modification a d'abord été effectuée par T. Johansson et F. Jönsson [7] dans le contexte d'un décodage à maximum de vraisemblance. On reprendra exactement les mêmes notations que précédemment. On considère une fonction : $\mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ telle que

$$\Delta(f,RM[1,m]) \leq \frac{1}{2} - \epsilon.$$

Rappelons que si $C_i(x_1,\ldots,x_i)$ désigne une fonction affine alors on note (cf. formule 6.1):

$$\mathcal{E}^{(f)}(C_i) = 1 - 2^{-m+i} \sum_{l=0}^{2^{m-i}-1} M(f(x_1,\ldots,x_i,\overline{s_j}),C_i(x_1,\ldots,x_i)).$$

On a montré que si C_i est une fonction affine dont les coefficients sont les i premiers coefficients d'une fonction affine C vérifiant

$$\Delta(f,C) \leq \frac{1}{2} - \epsilon,$$

alors

$$\mathcal{E}^{(f)}(C_i) \geq \frac{1}{2} + \epsilon.$$

Le principe de l'algorithme de Goldreich et Levin consiste à accepter un candidat C_i si, après un nombre $\mathcal{O}(1/\epsilon)$ d'essais on finit par trouver un élément $\overline{s} \in \mathbb{F}_2^{m-i}$ tel que

$$M(f(x_1,\ldots,x_i,s),C_i(x_1,\ldots,x_i)) \leq \frac{1}{2} - \frac{2\epsilon}{3}.$$

Notre principe est différent, on sait que par définition $\mathcal{E}^{(f)}(C_i) \geq 1/2+\epsilon$ si C_i est un bon candidat. C'est une condition nécessaire. Notre principe sera de donner une approximation de la quantité $\mathcal{E}^{(f)}(C_i)$, c'est-à-dire que nous accepterons un candidat C_i si celui-ci satisfait :

$$\mathcal{E}^{(f)}(C_i) \geq \frac{1}{2} + \epsilon - \frac{\epsilon}{c}, \quad c > 1,$$

où ϵ/c représente l'écart type de l'erreur que l'on s'autorisera. Nous avons représenté notre programme d'approximation à la ième étape de notre algorithme à la figure 6.6.

```
Approx-candidat : = fonction(f,ε,m,c,u,(c_1,...,c_i))

t :=^def poly(m/ε,c); (= poly(ε,c,u cf. théorème 32)
t' :=^def poly(m/ε,c); (= poly(ε,c,u cf. théorème 32)
compt := 0;

   pour v de 1 à t faire
    {
      Prendre s̄ = (s_{i+1},...,s_m) ∈ F_2^{m-i};
      h := 0;
      pour k de 1 à t' faire
       {
         Prendre r̄ = (r_1,...,r_i) ∈ F_2^i;
         h := h + [f(r̄,s̄) − Σ_{j=1}^{i} c_j r_j];
       }
       compt := compt + min(t' − h,h);
    }

retourner compt;
```

FIG. 6.6 – *Approximation à la i-ième étape*

Si $t = 2^{m-i}$ et $t' = 2^i$ sont les paramètres de notre programme d'approximation, alors ce programme calcule précisément $\mathcal{E}^{(f)}(C_i)$. Nous déterminerons par la suite les quantités optimales du point de vue de la complexité pour t et t'. Le programme final qui va reconstruire la ou les solutions linéaires de notre problème se trouve à la figure 6.7, on peut d'ailleurs faire la même remarque que pour l'algorithme de Goldreich-Levin en ce qui concerne les premiers coefficients des fonctions affines qui répondent à notre problème de reconstruction. Ces coefficients se détermine avec une simple estimation de la quantité $\Delta(f,C)$ où C est une fonction linéaire satisfaisant $M(f,C) \leq 1/2 - \epsilon$.

On va maintenant étudier la complexité de cet algorithme, on fixera les paramètres t,t' de l'algorithme *"Approx-candidat"* selon le type canal dans lequel on travaillera.

Théorème 34. *Soient $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ tel que $\Delta(f,RM[1,m]) \leq \frac{1}{2} - \epsilon$. Alors il y a au plus $2/\epsilon^3$ fonctions affines $C_i(x_1,\ldots,x_i) = c_0 + c_1x_1 + \ldots + c_ix_i$, $i \in [1\ldots m]$, telles que*

$$\mathcal{E}^{(f)}(C_i) \geq \frac{1}{2} + \epsilon,$$

où $\mathcal{E}^{(f)}(C_i)$ est la quantité définie par 6.1.

Preuve : Rappelons que

$$\mathcal{E}^{(f)}(C_i) = 1 - 2^{-m+i} \sum_{l=0}^{2^{m-i}-1} M(f(x_1,\ldots,x_i,\overline{s_l}),C_i(x_1,\ldots,x_i)).$$

Maintenant, on sait qu'il y a au plus $2/\epsilon^2$ fonctions affines C telles que

$$M(f(x_1,\ldots,x_i,\overline{s_j}),C) \leq 1/2 - \epsilon.$$

D'autre part on sait que l'on a

$$M(f(x_1,\ldots,x_i,\overline{s_j}),C_i) \leq 1/2 - \epsilon/2,$$

pour une fraction au moins égale à $\mathcal{O}(\epsilon)$ des $\overline{s_j}$. Donc

$$(\text{Taille max de liste}) \times \epsilon 2^{m-i} \leq 2^{m-i} \times \frac{2}{\epsilon^2}.$$

Donc la liste de candidats est d'au plus $2/\epsilon^3$. □

On va maintenant étudier cet algorithme suivant les deux canaux considérés, c'est-à-dire le canal binaire symétrique et le canal adversaire.

Le canal binaire symétrique

On suppose que l'on se donne une fonction affine C qui a été bruitée par un processus aléatoire, c'est-à-dire qu'une fraction des sorties de f ont été modifiées

```
Reconstruct-linear(f,ε,m,c,u)

   R := [0]; (c'est une liste de fonctions affines).
   L := [.]; (c'est une liste de fonctions affines).

 pour i de 1 à m faire
    {
      pour v de 1 à card(R) faire
        {
           L = L ∪ (R[v] + x_i);
        }

      R = {};

      pour v de 1 à card(L) faire
        {
          si Approx-candidat (f,ε,m,c,u,L[v]) ≥ 1/2 − ε/c alors
             R = R ∪ L[v];
        }

      L := R;
    }

 retourner L;
```

FIG. 6.7 – *Algorithme adaptatif de reconstruction*

et que la position de ces erreurs est uniformément distribuée. On notera donc f la fonction résultante, et on supposera donc que $\Delta(f,C) = \frac{1}{2} - \epsilon$. On va dans une première partie justifier la métrique M que l'on utilise. On a ici repris la démonstration de T. Johansson et F. Jönsson [7].

Dans ces conditions, on a que

$$\underset{x,y\in\mathbf{F}_2^m}{\mathbf{Prob}}\left(f(x)+f(y)=C(x+y)\right)=\frac{1}{2}+2\epsilon^2,$$

en généralisant, on obtient aisément que

$$\underset{y_j\in\mathbf{F}_2^m,\ j\in\{1,..,l\}}{\mathbf{Prob}}\left(\sum_{j=1}^{l} f(y_j)=C(\sum_{j=1}^{l} y_j)\right)=\frac{1}{2}+2^{l-1}\epsilon^l,$$

donc, si on note e la variable aléatoire telle que $C(\sum_{j=1}^{l} y_j) = \sum_{j=1}^{l} f(y_j) + e$ on a

$$\mathbf{Prob}(e = 0) = \frac{1}{2} + 2^{l-1}\epsilon^l.$$

Posons $C(x) = c_1x_1 + \ldots + c_mx_m$. Si $\sum_{j=1}^{l} y_j$ désigne l'élément $(r_1, \ldots, r_i, s_i) \in \mathbb{F}_2^m$, alors l'expression $C(\sum_{j=1}^{l} y_j) = \sum_{j=1}^{l} f(y_j) + e$ est équivalente à

$$\sum_{j=1}^{i} c_j r_j + \sum_{j=i+1}^{m} c_j s_j = f(r_1, \ldots, r_i, s) + e, \quad s = (s_{i+1}, \ldots, s_m) \in \mathbb{F}_2^{m-i}.$$

Soit $\hat{C}_i(x_1, \ldots, x_i) = \sum_{j=0}^{i} \hat{c}_j x_j$ un candidat suggéré, alors on a

$$\sum_{j=1}^{i} (c_j + \hat{c}_j) r_j + \sum_{j=i+1}^{m} c_j s_j + e = \sum_{j=1}^{l} f(y_j) + \sum_{j=1}^{i} \hat{c}_j r_j.$$

Maintenant, appelons W la variable aléatoire $\sum_{j=i+1}^{m} c_j s_j$. Supposons que l'on ait choisi le bon candidat $\hat{C}_i(x_1, \ldots, x_i) = \sum_{j=0}^{i} \hat{c}_j x_j$. Notons

$$num_s = \sum_{k=1}^{t'} \left[f(r_1^{(k)}, \ldots, r_i^{(k)}, s) - \sum_{j=1}^{i} \hat{c}_j \cdot r_j^{(k)} \right],$$

où la somme est prise sur $\mathbb{N}$, tandis que les expressions

$$\left[f(r_1^{(k)}, \ldots, r_i^{(k)}, s) - \sum_{j=1}^{i} \hat{c}_j \cdot r_j^{(k)} \right]$$

sont dans $\{0,1\}$. Rappelons que le programme *"Approx-candidat"* calcule précisement la quantité

$$\sum_{j=1}^{t} \max\left(t' - num_{s_j}, num_{s_j}\right) = \frac{tt'}{2} - \sum_{j=1}^{t} \frac{\left|t' - 2 \cdot num_{s_j}\right|}{2}, s_j \in \mathbb{F}_2^{m-i}$$

avec les même quantités t et t' de ce programme. Comme $\sum_{j=1}^{i} (c_j + \hat{c}_j) r_j = 0$, on a

$$\underset{(r^{(j)}, s^{(j)})}{\mathbf{Prob}} \left(\sum_{j=1}^{i} (c_j + \hat{c}_j) r_j + \sum_{j=i+1}^{m} c_j s_j + e = 0 \right) = \mathbf{Prob}(W + e = 0).$$

Cette probabilité est soit $1/2 - 2^{l-1}\epsilon^l$ ou bien $1/2 + 2^{l-1}\epsilon^l$ selon que $W = 0$ ou $W = 1$. Donc num_s obéit à une loi binomiale $Bin(t', 1/2 - 2^{l-1}\epsilon^l)$. Rappelons que l'on est dans le cas du canal binaire symétrique, dans ce cas on a vu dans la

preuve du théorème 29 que si $t' = \mathcal{O}(\epsilon^{-2})$ et donc $2^i = \mathcal{O}(\epsilon^{-2})$, alors num_s ne dépend pas de s. Donc dans la suite nous noterons num.

Si l'on a choisi un mauvais candidat on a $\sum_{j=1}^{i}(c_j + \hat{c}_j)r_j \neq 0$ et num obéit à la loi binomiale $Bin(t',1/2)$.

Nous allons maintenant justifier le choix de notre métrique que l'on a notée M. Notons H_0 l'événement : le candidat est correct, c'est-à-dire que ce candidat représente les i premiers coéfficients d'une fonction affine qui est solution de notre problème de reconstruction et H_1 l'événement : le candidat est incorrect, c'est-à-dire l'évènement contraire. Posons $p_0 = \mathbf{Prob}(e = 0) = \frac{1}{2} + 2^{i-1}\epsilon^i$. On sait que $\mathbf{Prob}(W = 0) = 1/2$. On a vu que num obéissait aux lois suivantes :

$$num|H_0 \in Bin(N,1/2),$$
$$num|H_1,W = 0 \in Bin(N,p_0),$$
$$num|H_1,W = 1 \in Bin(N,1 - p_0).$$

On peut naturellement approcher ces lois par des lois normales qui seront de bonnes approximations pour $t'p_0(1 - p_0) >> 10$. Soit la variable aléatoire $Y = |2 \cdot num - t'|$. Si $\mathbf{Prob}(2 \cdot num - t' < 0|H_1,W = 0) = \mathbf{Prob}(2 \cdot num - t' > 0|H_1,W = 1)$ est petit alors Y admet les densités suivantes :

$$f_{Y|H_0}(y) = \frac{2}{\sqrt{\pi t'}}e^{-y^2/t'}$$
$$f_{Y|H_1}(y) = \frac{2}{\sqrt{4\pi t' p_0(1-p_0)}}e^{-\frac{(y-t'p_0)^2}{4t'p_0(1-p_0)}}$$

Le meilleur candidat sera tel que la quantité

$$\Lambda = \frac{\mathbf{Prob}(H_1|Y)}{1 - \mathbf{Prob}(H_1|Y)} = \frac{\mathbf{Prob}(H_1|Y)}{\mathbf{Prob}(H_0|Y)} = \frac{\mathbf{Prob}(Y|H_1)\mathbf{Prob}(H_1)}{\mathbf{Prob}(Y|H_0)\mathbf{Prob}(H_0)}$$

est maximale. Dans l'article de Johansson-Jönsson [7], il est prouvé que

$$\Lambda = (ap_0(1 - p_0))^{t'/2}\cosh(\log(\frac{p_0}{1 - p_0})|\frac{t'}{2} - num|) \cdot \frac{\mathbf{Prob}(H_1)}{\mathbf{Prob}(H_0)}.$$

Par commodité, on va déterminer

$$\max_{(c_1,\ldots,c_i)}[\log(\Lambda)],$$

ce qui revient à déterminer le maximum de la fonction

$$\log\left(\cosh(\log(\frac{p_0}{1 - p_0})|\frac{t'}{2} - num|)\right),$$

on peut donner l'équivalent suivant :

$$\log(\cosh x) \approx \begin{cases} |x| - \log 2 & \text{si } |x| > 1, \\ \frac{x^2}{2} & \text{si } |x| < 1. \end{cases}$$

Donc si $|\frac{t'}{2} - num| > e$, la métrique $|.|$ est optimale. Si $|\frac{t'}{2} - num| < e$ alors la métrique $(.)^2$ est optimale. Cette démonstration est valable pour une valeur de $\overline{s}$. Ces raisonnements tiennent pour un candidat $\overline{s}$; pour t valeurs de $\overline{s}$, remarquons que l'on a

$$\mathbf{Prob}(Y|H_0) = \mathbf{Prob}(Y_1|H_0)\mathbf{Prob}(Y_2|H_0)\ldots\mathbf{Prob}(Y_t|H_0),$$

en conséquence la distance résultante utilisée sera:

$$dist = |2 \cdot num_1 - t'| + |2 \cdot num_2 - t'| + \ldots + |2 \cdot num_n - t'|,$$

ou bien

$$dist = (2 \cdot num_1 - t')^2 + (2 \cdot num_2 - t')^2 + \ldots + (2 \cdot num_n - t')^2.$$

Ce raisonnement tient à partir de la i-ième variable, avec $2^i = \mathcal{O}(\epsilon^{-2})$, c'est-à-dire pour $t' = \mathcal{O}(\epsilon^{-2})$. Donc

$$\left|\frac{t'}{2} - num\right| = t'\left|\frac{1}{2} - \frac{num}{t'}\right| \approx |\epsilon^{-1}|.$$

Si $|\epsilon^{-1}| > e$, c'est donc la métrique $|.|$ qu'il faut utiliser, ce qui correspond à une translation près, à notre métrique M. Dans le cas contraire c'est la métrique carrée qui est la meilleure.

A partir de maintenant, on va considérer notre métrique M pour analyser la complexité de notre algorithme puisqu'elle est bien plus adaptée aux problèmes que l'on considère.

Théorème 35. *Soient $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ tel que $\Delta(f,RM[1,m]) \leq \frac{1}{2} - \epsilon$. Supposons que la fonction f est équivalente à une fonction affine pour laquelle on aurait perturbé aléatoirement une fraction d'au plus $\frac{1}{2} - \epsilon$ de ses images. Supposons de plus que le nombre de variables soit si grand $(2^m = \mathcal{O}(\epsilon^{-2}))$, qu'il n'y ait qu'une seule fonction affine C qui vérifie $\Delta(f,C) \leq 1/2 - \epsilon$. Alors le problème de reconstruction à clairs choisis peut être résolu avec une complexité de l'ordre de*

$$\mathcal{O}\left((\log(2m) + u)\left(\frac{\log^2(1/\epsilon^2)}{\epsilon^4} + \frac{(m - \log(1/\epsilon^2))^2}{\epsilon^2}\right)\right)$$

opérations binaires et avec une probabilité d'échec inférieur à 2^{-u} en utilisant notre algorithme adaptatif de reconstruction 6.7.

Preuve : Reprenons la formule 6.3 et déterminons le nombre d'étapes nécessaires pour obtenir un décodage unique avec une probabilité supérieur à $1 - 2^{-v}$:

$$1 - R_C\left(\lfloor i(1/2 - \epsilon')\rfloor\right) \lesssim \frac{1}{\epsilon'}\sqrt{\frac{i}{\pi}}e^{-\frac{i\epsilon'^2}{4}} \leq 2^{-v},$$

ce qui implique assymptotiquement que

$$2\log(i) - 2\log(\pi) + 4v\log(2) - 4\log(\epsilon') \leq i\epsilon'^2,$$

donc si

$$i > \frac{-2\log(\pi) + 4v\log(2) - 4\log(\epsilon')}{\epsilon'^2},$$

alors $1 - R_C(\lfloor i(1/2 - \epsilon')\rfloor) \leq 2^{-v}$. Il faut maintenant déterminer la taille de l'échantillon nécéssaire pour évaluer $M(f,C)$. Soit $X_i = f(x,s_i) - C(x,s_i)$ la variable aléatoire qui est égale à 0 avec une probabilité $1/2 + \epsilon$. L'inégalité de Hoeffding nous dit que:

$$\textbf{Prob}\left[\left|\frac{X_1 + \ldots + X_N}{N} - \left(\frac{1}{2} + \epsilon\right) \geq \frac{\epsilon}{c}\right|\right] \leq 2^{1 - \frac{2N\epsilon^2}{c^2\log(2)}}.$$

On veut une probabilité d'échec qui décroît exponentiellement avec la taille de l'échantillon, soit:

$$2^{1 - \frac{2N\epsilon^2}{c^2\log(2)}} \leq 2^{-w}.$$

Donc si $N \geq c^2(\log(2) + w)/\epsilon^2$, cette condition est satisfaite. Cette inégalité nous dit deux choses, si $2^i \geq c^2(\log(2) + w)/\epsilon^2$, alors comme on est dans les conditions du canal binaire symétrique: $M(f(x,s_i),C(x,s_i)) \leq 1/2 - \epsilon \pm \epsilon/c$ avec probabilité supérieur à $1 - 2^{-w}$ et la formule 6.3 implique que $C(x,s_i)$ est l'unique fonction linéaire qui vérifie $M(f(x,s_i),C(x,s_i)) \leq 1/2 - \epsilon \pm \epsilon/c$. D'autre part l'inégalité de Hoeffding donne la taille d'échantillon nécéssaire pour évaluer cette distance la probabilité de succés que l'on souhaite. Comme il y a m étapes il faut évaluer la probabilité que l'évènement $FV(C)$ "une distance $M(f,C)$ est mal évaluée durant ces m étapes", soit:

$$m2^{1 - \frac{2N\epsilon^2}{c^2\log(2)}}.$$

Donc si $N > c^2(\log(2m) + w')/\epsilon^2$, la probabilité d'une distance mal évaluée durant les m étapes est inférieur à $2^{-w'}$. Pendant ces i premières étapes, on a 2^i candidats à traiter.la liste des candidats peut en effet croître pour atteindre une taille de l'ordre $\mathcal{O}(\epsilon^{-2})$. On peut maintenant fixer correctement les paramètres t,t' du programme "Test-candidat". Soient $t = \mathcal{O}(1)$ et $t' = \mathcal{O}(c^2(\log(2m) + w')/\epsilon^2)$, donc on va pouvoir maintenant donner la complexité de l'algorithme. Pour les i première étapes on obtient:

$$\mathcal{O}\left((\log(2m) + u)\frac{\log^2(1/\epsilon^2)}{\epsilon^4}\right).$$

Puis pour les $m - i$ restantes, on a montré la liste des candidats potentiels se limitait à un unique candidat, donc en déduit la complexité:

$$\mathcal{O}\left((\log(2m) + u)\frac{(m - \log(1/\epsilon^2))}{\epsilon^2}\right).$$

Donc le nombre d'opérations binaires à effectuer pour retourner la liste des fonctions affines est inférieur ou égal à

$$\mathcal{O}\left((\log(2m)+u)\left(\frac{\log^2(1/\epsilon^2)}{\epsilon^4}+\frac{(m-\log(1/\epsilon^2))^2}{\epsilon^2}\right)\right)$$

pour une probabilité d'échec inférieure à 2^{-u}.

□

6.3.4 Utilisation de la transformée de Fourier rapide

La complexité de cet algorithme est intéressante, mais on va cependant modifier notre approche pour obtenir les k ($2^k = \mathcal{O}(1/\epsilon^2)$) premiers coefficients de la fonction affine C satisfaisant $\Delta(f,C) \leq 1/2 - \epsilon$, on va utiliser la transformée de Fourrier rapide. Rappelons que la transformée de Fourier d'une fonction $g : \mathbb{F}_2^k \rightarrow \mathbb{F}_2$ fonction g en $(c_1,...,c_k) \in \mathbb{F}_2^k$ est noté $\hat{F}_g(c_1,...,c_k)$, et que l'on a la relation

$$d(g,c_1x_1 + ... + c_kx_k) = \frac{1}{2}(2^k - \hat{F}_g(c_1,...,c_k)).$$

Maintenant prenons un élément $s \in \mathbb{F}_2^{m-k}$, alors si on pose

$$g(x_1,...,x_k) = f(x_1,...x_k,s),$$

on a de même

$$d(g,c_1x_1 + ... + c_kx_k) = \frac{1}{2}(2^k - \hat{F}_g(c_1,...,c_k)).$$

Maintenant pour obtenir les k premiers coefficients C, il suffit, comme dans l'algorithme 6.7 d'effectuer les calculs de

$$\sum_{i=1}^{T} M(f(x_1,...x_k,s_i),c_1x_1 + ... + c_kx_k),\ s \in \mathbb{F}_2^{m-k}$$

pour tout $(c_1,...,c_k) \in \mathbb{F}_2^k$ où $T = \mathcal{O}(1)$ dans le canal binaire symétrique, comme nous l'avons montré dans la preuve du théorème 32. Enfin rappelons qu'avec un développement diadique on identifie les fonctions affines du type $\mathbb{F}_2^k \rightarrow \mathbb{F}_2$ avec les éléments de $\mathbb{Z}$. On trouvera donc un programme qui calcule les k premiers coefficients de la fonction linéaire C à la figure 6.8.

On va maintenant donner un algorithme adaptatif de reconstruction qui utilise des transformées de Fourier rapides pour calculer les k premiers coefficients de la ou les fonctions linéaires les plus proches de f pour la métrique M et le seuil de $1/2 - \epsilon$ que l'on a défini par exemple dans le théorème 32.

Théorème 36. *Soient $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ tel que $\Delta(f,RM[1,m]) \leq \frac{1}{2} - \epsilon$. Supposons que la fonction f est équivalente à une fonction affine pour laquelle on aurait*

```
FFT-Reconstruct-linear(f,ε,m,c,T,u)

    R := {0}; (ce sont des listes de fonctions affines).
    E := {}; (c'est une liste d'entiers).
    F := {0,...,0}; (c'est une liste d'entiers).
    compt := 0; (c'est un compteur).
    g; une fonction
    cte := partie entière de O(uc²/ε²);
    lcte := partie entière de log₂(cte);

  pour j de 1 à T faire
      {
        Prendre aléatoirement s ∈ F_2^{m-lcte};
        g := f(x_1,...,x_k,s);
        E[i] := M(g,i) pour i ∈ [1,...,cte];

          pour l de 1 à cte faire
            F[l] := F[l] + E[l];
      {
  pour i de 1 à cte faire
      {
        si F[l] ≤ T × 1/2 − ε + ε/c;
         {
            R[compt] := i;
            compt := compt + 1;
         }
      {

retourner R;
```

FIG. 6.8 – *Programme de calcul des premiers coefficients*

perturbé aléatoirement une fraction d'au plus $\frac{1}{2} - \epsilon$ *de ses images. Supposons de plus que le nombre de variables soit si grand* ($2^m = \mathcal{O}(\epsilon^{-2})$), *qu'il n'y ait qu'une seule fonction affine* C *qui vérifie* $\Delta(f,C) \leq 1/2 - \epsilon$. *Alors le problème de reconstruction à clairs choisis dans le canal binaire symétrique peut être résolu avec une complexité de l'ordre de*

$$\mathcal{O}\left((\log(2m) + u)\frac{m^2}{\epsilon^2}\right)$$

opérations binaires et avec une probabilité d'échec inférieure à 2^{-u} avec l'algorithme adaptatif rapide de reconstruction 6.9.

Preuve : Nous allons reprendre la preuve du théorème précédent, simplement on va déterminer les k premiers coefficients de la fonction linéaire C qui satisfait $M(f,C) \leq 1/2 - \epsilon$ d'une autre manière. Soit $s \in \mathbb{F}_2^{m-k}$. Notons $g(x_1,\ldots,x_k) = f(x_1,\ldots,x_k,s)$, et $F(v) = (-1)^{f(v)}$ pour $v \in \mathbb{F}_2^k$, alors si $\hat{F}$ désigne la transformée de Hadamard de F, on a vu que

$$d(g,c_1x_1 + \ldots + c_kx_k) = \frac{1}{2}(2^k - \hat{F}(c_1,\ldots,c_k)),$$

et on a aussi vu que le coût du calcul de $\hat{F}$ était de $\mathcal{O}(k2^k)$ opérations binaires. Comme on suppose que l'on est déjà à la k-ième étape de notre algorithme on a juste à effectuer le calcul de

$$\frac{1}{T}\sum_{i=1}^{T} M(f(x_1,\ldots,x_k,s_i),c_1x_1 + \ldots + c_kx_k),\ T = \mathcal{O}(1),$$

et comme

$$M(f,c_1x_1 + \ldots + c_kx_k) = \min(d(g,c_1x_1 + \ldots + c_kx_k),1 - d(g,c_1x_1 + \ldots + c_kx_k)),$$

on en déduit que le coût de ce calcul sera de l'ordre de $\mathcal{O}(k2^k) = \mathcal{O}(k/\epsilon^2)$ opérations binaires. Il reste ensuite à faire le tri pour ne garder que les fonction affines $c_1x_1 + \ldots + c_kx_k$ qui satisfont

$$\frac{1}{T}\sum_{i=1}^{T} M(f(x_1,\ldots,x_k,s_i),c_1x_1 + \ldots + c_kx_k) \leq \frac{1}{2} - \epsilon.$$

Le reste des coefficients s'obtient de la même manière que dans le théorème précédent. On peut donc en déduire la complexité finale de cet algorithme,

$$\mathcal{O}\left((\log(2m) + u)\frac{m^2}{\epsilon^2}\right)$$

opérations binaires et avec une probabilité d'échec inférieure à 2^{-u}. □

Remarque 15. *Nous présenterons nos résultats de simulation à la fin de cette section, ainsi nous comparerons les différents algorithmes.*

Le canal adversaire

Dans ce cas on ne peut pas appliquer les règles classiques d'échantillonage en fixant la valeur des $m - i$ dernières variables, par exemple soit i tel que $2^i \gg \frac{1}{\epsilon^2}$, alors il est tout à fait possible de construire une fonction f telle que

```
Reconstruct-linear-fast(f,ε,m,c,u)

   R := {0}; (c'est une liste de fonctions affines).
   L := {}; (c'est une liste de fonctions affines).
   cte := partie entière de O(uc²/ε²);
   lcte := partie entière de log₂(cte);

  R := FFT-Reconstruct-linear(f,ε,m,c,O(1),u);

 pour i de lcte + 1 à m faire
    {
      pour v de 1 à card(R) faire
        {
           L = L ∪ (R[v] + x_i);
        }

      R = {};

      pour u de 1 à card(L) faire
        {
          si Approx-candidat (f,ε,m,L[u]) ≤ 1/2 − ε/c alors
             R = R ∪ L[u];
        }

      L := R;
    }

 pour j de 1 à card(R) faire
    {
      si (Δ(f,R[j]) > 1/2) alors
         R[j] := R[j] + 1;
    }
 retourner R;
```

FIG. 6.9 – *Algorithme adaptatif rapide de reconstruction*

$\Delta(f(x_1,\ldots,x_m),RM_1[1,m]) < \frac{1}{2} - \epsilon$ avec $\Delta(f(x_1,\ldots,x_i,\overline{s_t}),RM_1[1,i]) > \frac{1}{2} - \epsilon$ pour certaine valeurs de t il suffit pour cela de placer les erreurs dans une région particulière de l'espace $\mathbb{F}_2^m$.

Théorème 37. *Soient $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ tel que $\Delta(f,\epsilon) \leq \frac{1}{2} - \epsilon$. Alors le problème de reconstruction à clairs choisis dans le canal adversaire peut être résolu avec une complexité de l'ordre de*

$$\mathcal{O}\left(\frac{(u+\log(4m/\epsilon^2))^2 m^2}{\epsilon^7}\right)$$

opérations binaires et avec une probabilité d'échec inférieure à 2^{-u} avec l'algorithme 6.7.

Preuve : Soit $X(s_i) = M(f(x,s_i),C_i(x))$ une variable aléatoire telle que $1/2 \leq X_{s_i} \leq 1$, avec $s_i \in \mathbb{F}_2^{m-i}$. On veut donner une approximation de ces N' variables aléatoires indépendantes ($0 \leq i \leq N'$) avec une précision α. L'inégalité de Hoeffding nous donne

$$\textbf{Prob}\left[\left|\frac{S_N(s_i)}{N} - X(s_i)\right| > \alpha\right] \leq 2^{1-\frac{2N\alpha^2}{\log(2)}}.$$

Donc pour N' valeurs de s_i, on prendra l'union des évènements. Finalement la probabilité pour qu'une de ces variables aléatoires soit mal approximée est inférieure à $N'2^{1-\frac{2N\alpha^2}{\log(2)}}$. D'autre part cette opération sera effectuée m fois car l'algorithme fonctionne en m étapes et il ne faut pas oublier qu'il y a au plus $1/\epsilon^2$ solutions a notre problème. Donc la probabilité qu'un bon candidat soit mal approximé au cours de ces m étapes est inférieure à

$$\frac{mN'}{\epsilon^2} 2^{1-\frac{2N\alpha^2}{\log(2)}}.$$

Maintenant on cherche à approximer $\mathcal{E}^{(f)}(C_i)$. On a vu dans la preuve du lemme 16 que si $C_i = c_1x_1 + \ldots + c_ix_i$ représentait les premiers coefficients d'une solution $C(x_1,\ldots,x_m) = c_0 + \ldots + c_mx_m$ de notre problème alors

$$\mathcal{E}^{(f)}(C_i) \geq 1 - M(f,C) \geq \frac{1}{2} + \epsilon.$$

On va supposer que l'on connaît avec exactitude la quantité

$$X(s_i) = M(f(x,s_i),C_i(x)).$$

Les éléments $s_i \in \mathbb{F}_2^{m-i}$ sont pris aléatoirement, donc ils sont indépendants, et par suite les $X(s_i)$ sont indépendants. On peut donc appliquer l'inégalité de Hoeffding à ces variables pour obtenir la probabilité qu'un candidat soit mal approximé à la i-ème étape de l'algorithme. Soit $\mu_i = \mathcal{E}^{(f)}(C_i) \geq 1/2 + \epsilon$, alors on a:

$$\textbf{Prob}\left[\left|\frac{X(s_1) + \ldots + X(s_{N'})}{N'} - \mu_i\right| > \beta\right] \leq 2^{1-\frac{2N'\beta^2}{\log(2)}}.$$

Maintenant comme il y a m étapes et au plus $1/4\epsilon^2$ solutions à notre problème, on en déduit la probabilité d'échec :

$$\frac{m}{\epsilon^2}2^{1-\frac{2N'\beta^2}{\log(2)}}.$$

On va pouvoir en déduire la probabilité qu'un bon candidat C_i soit rejeté. On a :

$$\mathcal{E}^{(f)}(C_i) - \frac{\sum_{j=1}^{N'} \frac{S_N(s_j)}{N}}{N'} = \mathcal{E}^{(f)}(C_i) - \frac{X(s_1) + \ldots + X(s_{N'})}{N'} +$$

$$\frac{X(s_1) + \ldots + X(s_{N'})}{N'} - \frac{\sum_{j=1}^{N'} \frac{S_N(s_j)}{N}}{N'}.$$

Donc la probabilité qu'un candidat soit rejeté au cours des m étapes est inférieure à :

$$\frac{mN'}{\epsilon^2}2^{1-\frac{2N\alpha^2}{\log(2)}} + \frac{m}{\epsilon^2}2^{1-\frac{2N'\beta^2}{\log(2)}}.$$

Posons $N = N'$ et $\beta = \alpha = \epsilon/c$, alors on a :

$$\frac{mN'}{\epsilon^2}2^{1-\frac{2N\alpha^2}{\log(2)}} + \frac{m}{\epsilon^2}2^{1-\frac{2N'\beta^2}{\log(2)}} \leq \frac{2mN}{\epsilon^2}2^{1-\frac{2N\alpha^2}{\log(2)}}.$$

on va donc chercher N tel que

$$\frac{2mN}{\epsilon^2}2^{1-\frac{2N\alpha^2}{\log(2)}} \leq 2^{-u}.$$

Si :

$$N > \frac{c^2(u + \log(4m/\epsilon^2))}{2\epsilon^2},$$

la probabilité qu'un bon candidat soit rejeté est inférieure à 2^{-u}. N et N' sont fixés donc on va pouvoir donner la complexité de cet algorithme. On a montré que la liste des candidats pouvait atteindre $2/\epsilon^3$, donc on obtient une complexité de l'ordre de

$$\frac{c^4(u + \log(4m/\epsilon^2))^2 m^2}{\epsilon^7},$$

avec une probabilité d'échec inférieure à 2^{-u}. □

6.3.5 Quelques résultats expérimentaux

Dans le canal binaire symétrique

On a construit une fonction :

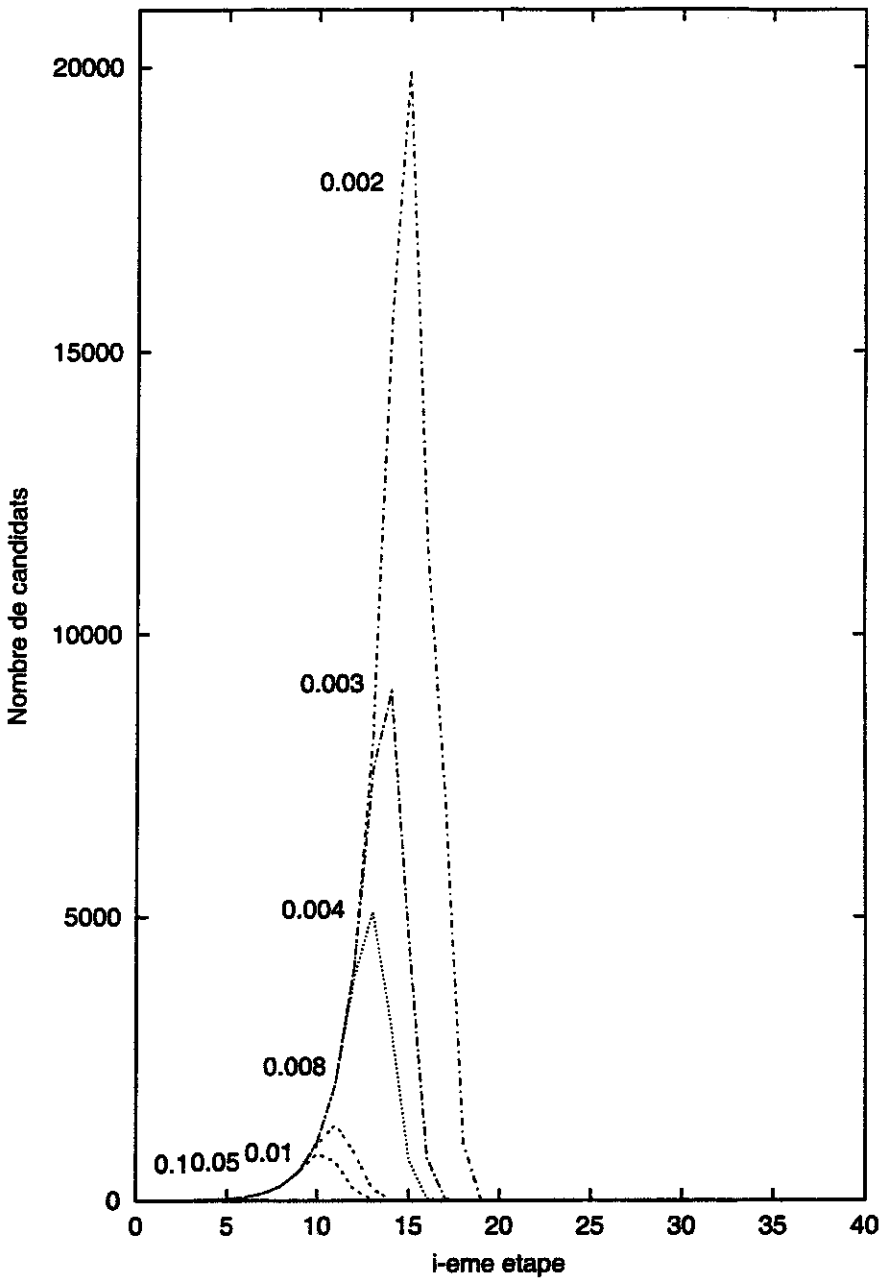
$$f : \mathbb{F}_2^{64} \longrightarrow \mathbb{F}_2$$

qui coïncide avec la fonction linéaire

$$C : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2,\ C(x_1,\ldots,x_{64}) = x_7 + x_{13} + x_{19} + x_{25} + x_{33} + x_{40} + x_{62}$$

sur une fraction $\frac{1}{2} + \epsilon$ de ces entrées. Les images de cette fonction ont été perturbées par un processus aléatoire. On a implémenté cette fonction d'une telle manière que l'on peut faire varier le paramètre ϵ. On a montré qu'au cours des étapes de l'algorithme de la figure 6.7, la liste des candidats potentiels est bornée par la borne de Johnson. On va donc voir effectivement comment évolue cette liste au cours de l'algorithme *"Reconstruct-linear"* 6.7. On a représenté graphiquement le nombre de candidats en fonction du nombre d'étapes de l'algorithme, c'est-à-dire encore du nombre de coefficients que l'on a déterminé. Chaque courbe est associée à un paramètre ϵ que l'on a représenté à côté de chacune d'entre elles. On a représenté les 40 premières étapes de l'algorithme, car durant les 24 dernières, seul un unique candidat est resté en liste. On a pas représenté la courbe pour l'algorithme de Goldreich-Levin car cet algorithme est plus limité du point de vue complexité, on a pu observer cependant que la liste des candidats augmentait plus au cours des étapes de l'algorithme de Goldreich-Levin. La variante de notre algorithme, utilisant la transformée de Fourrier rapide, n'a pas non plus été représentée car, dans ce cas-là, le nombre de candidats dans la liste est toujours

égal à un, pour les paramètres que l'on considère.



Les 40 premières étapes de l'algorithme 6.7

Cette expérience a été répétée 1000 fois pour $\epsilon \geq 0.008$, c'est-à-dire que la position des erreurs des image de notre fonction affine a été modifié 1000 fois, et à chaque fois, on a obtenu une courbe semblable. Le nombre de candidats n'a pas varié de façon notable par rapport à la courbe ci-dessus. Avec les paramètres que l'on considère notre algorithme n'a jamais retourné plus d'une solution.

On va maintenant donner les performances des trois algorithmes qu'on a analysé. On a utilisé la même fonction que précédemment. On a représenté un tableau indiquant le temps en secondes en fonction du paramètre ϵ et de l'algorithme que l'on considère, c'est-à-dire l'algorithme de Goldreich et Levin, notre algorithme et notre algorithme utilisant l'option de la transformée de Fourrier rapide. Par mesure de clarté, on a converti ϵ en terme de pourcentage de bruit. Ces résultats ont été obtenus avec un ordinateur P4 à 1.5 GHz.

ϵ	% noise	sans FFT	avec FFT	Gold.-Levin
0.40	10	0	0	63
0.35	15	0	0	144
0.30	20	0	0	207
0.25	25	0	0	339
0.20	30	0	0	703
0.15	35	0	0	6471
0.10	40	0	0	14467
0.09	41	0	0	?
0.08	42	0	0	?
0.07	43	0	0	?
0.06	44	0	0	?
0.05	45	0	0	?
0.04	46	1	0	?
0.03	47	3	1	?
0.02	48	8	3	?
0.015	48.5	15	7	?
0.01	49	34	14	?
0.007	49.3	69	27	?
0.006	49.4	87	29	?
0.005	49.5	141	53	?
0.004	49.6	203	60	?
0.003	49.7	397	115	?
0.002	49.8	1063	233	?
0.001	49.9	8861	894	?

Le facteur implémentation peut bien sûr faire varier les temps de calcul, mais nous n'avons pas réussi a obtenir de meilleurs temps pour l'algorithme de Goldreich-Levin.

Le canal adversaire

Le temps de calcul pour ce type de canal a moins d'intérêt car il dépend énormément de la fonction considérée. Pour ce canal, on a construit des exemples de fonctions qui sont difficilement approximables du point de vue du décodage par des fonctions affines. Elles ne sont pas forcément éloignées des fonctions affines, mais leur structure est plus compliquée du point de vue du décodage, c'est-à-dire, du point de vue de notre algorithme que la répartition des erreurs n'est pas uniformément distribuée. On a construit un exemple de fonction qui possède ce type de propriété. Pour construire cette fonction, on a simplement pris des monômes de bas degré, choisis aléatoirement :

$$\begin{aligned} f_1(x_1,\ldots,x_{64}) = \; & x_{27}x_{38} + x_{28}x_{59} + x_{27}x_{58} + x_{26}x_{57} + x_{24}x_{57} + \\ & x_5x_{17}x_{18} + x_6x_{18}x_{19} + x_7x_{19}x_{20} + x_8x_{20}x_{21} + x_9x_{21}x_{22} + \\ & x_{10}x_{22}x_{23} + x_{13}x_{25}x_{26} + x_{14}x_{26}x_{27} + x_{15}x_{27}x_{28} + x_{13}x_{56}x_{57} + \\ & x_{14}x_{58}x_{59} + x_{15}x_{59}x_{60} \end{aligned}$$

Voici pour cette fonction f_1 la liste des fonctions linéaires pour $\epsilon = 0.007$:

$x_{13} + x_{24} + x_{26}$
$x_{13} + x_{24} + x_{26} + x_{27}$
$x_{13} + x_{24} + x_{26} + x_{27} + x_{59}$
$x_{13} + x_{24} + x_{26} + x_{57} + x_{59}$
$x_{13} + x_{24} + x_{26} + x_{27} + x_{57} + x_{59}$
$x_{15} + x_{28} + x_{38} + x_{58} + x_{59}$
$x_{15} + x_{28} + x_{38} + x_{57} + x_{58} + x_{59}$

On peut aussi construire d'autres fonctions de manière systématique, et qui ont les mêmes propriétés par rapport à $RM[1,m]$. Soient k fonctions affines distinctes $h_i : \mathbb{F}_2^m \to \mathbb{F}_2$. Maintenant, considérons les éléments de $\mathbb{F}_2^m$ comme les entiers compris entre 0 et $2^m - 1$. Maintenant, on va construire notre fonction f de la manière suivante :
pour tout $x_1 + 2x_2 + \ldots + 2^{m-1}x_m \in \left[i \times \frac{2^m}{k}, (i+1) \times \frac{2^m}{k}\right]$ on pose

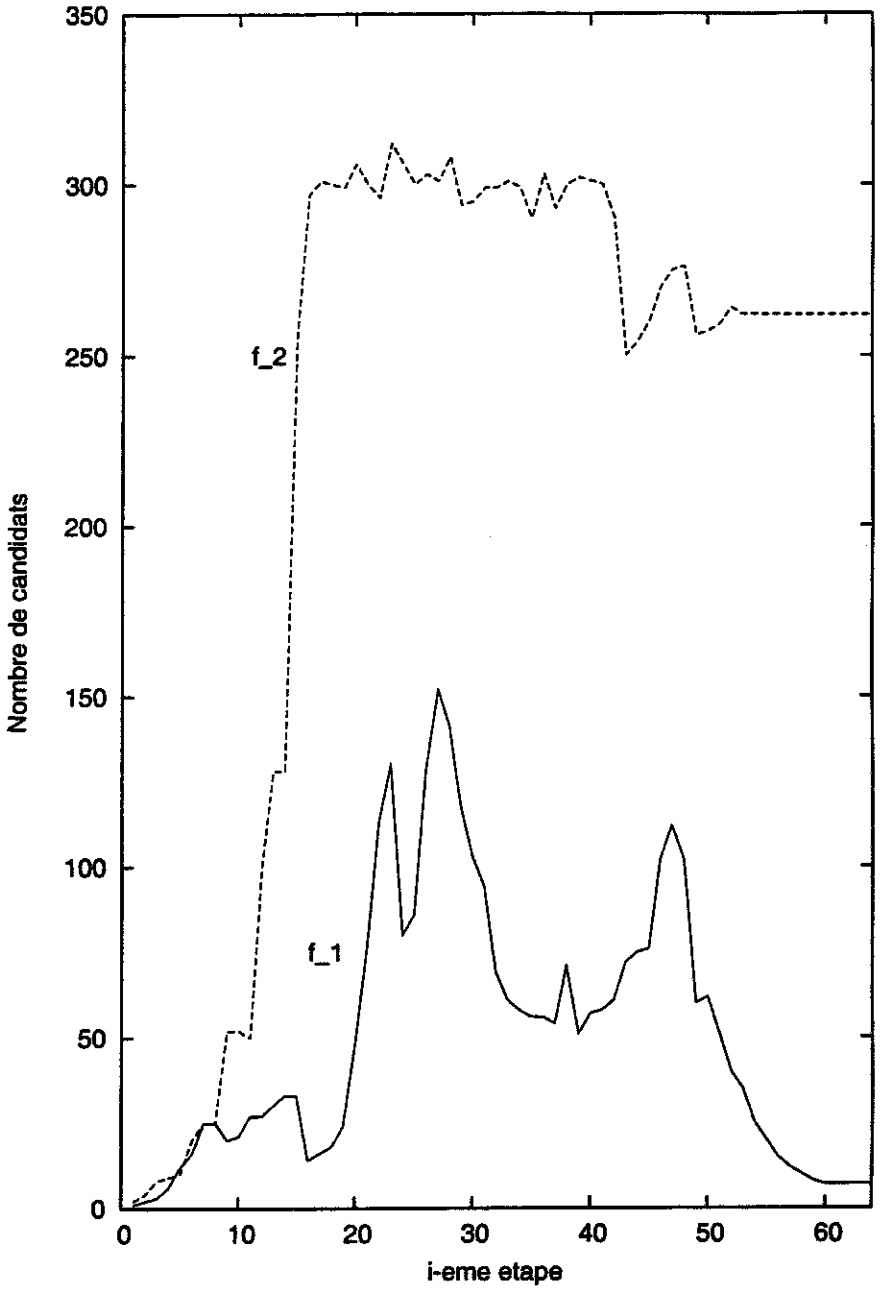
$$f(x_1,\ldots,x_m) = h_i(x_1,\ldots,x_m), \;\; i \in [1,\ldots,k].$$

Par construction, on a donc k fonction h_i qui vérifient

$$M(f,h_i) \leq \frac{1}{2} + \frac{1}{k}.$$

Nous avons construit une fonction de cette manière, avec $m = 64$ et $k = 34$ et nous l'avons appelé f_2. On peut construire de cette manière, ou de manière intuitive de nombreux exemples de fonctions pour lesquelles il existe plusieurs fonctions affines relativement proches de cette fonction, mais il est aussi très

intéressant d'observer comment évolue la taille de la liste au cours des étapes de notre algorithme. On a vu que dans le cas du canal binaire symétrique le comportement de cette liste était très régulier. Voici cette évolution pour les fonctions f_1 et f_2 pendant les 64 premières étapes.



On observe le caractère plus aléatoire que dans le cas du canal binaire symétrique de la variation du nombre de candidats.

6.4 Extension aux Reed-Muller d'ordre supérieur

Il est assez naturel de vouloir étendre les résultats obtenus pour les Reed-Muller d'ordre un aux Reed-Muller d'ordre supérieur. On s'est donc intéressé a ce type de décodage, mais dans des conditions particulières.

6.4.1 Les problèmes considérés

On va étudier le problème du décodage des Reed-Muller d'ordre deux dans le cas du canal binaire symétrique, on va étudier des algorithmes de décodage par liste. On s'est volontairement limité cette étude aux Reed-Muller d'ordre 2 pour des raisons de complexité. Les algorithmes que l'on va présenter s'étendent naturellement aux Reed-Muller d'ordre supérieur à deux, mais on verra que la complexité de ces algorithmes croit exponentiellement avec l'ordre du Reed-Muller considéré. On fera une étude théorique de ces algorithmes pour le décodage en grande longueur. On verra expérimentalement les performances en grande longueur, mais aussi en petite longueur. Nous expliquerons les problèmes que nous avons rencontrés pour étendre ces algorithmes au canal adversaire. En résumé, on se donne donc une fonction $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$ telle que $\Delta(f,RM[2,m]) \leq \frac{1}{2} - \epsilon$. On suppose que la fonction f est obtenue à partir d'une fonction quadratique dont on aurait perturbé, par un processus aléatoire, une fraction $\frac{1}{2} - \epsilon$ de ses images. On cherche donc la liste de toutes les fonctions quadratiques Q telles que $\Delta(f,Q) \leq \frac{1}{2} - \epsilon$.

6.4.2 Quelques notations

On utilisera les même notation que dans les chapitres précédents pour désigner les distances. On aura ici, avec la construction de Plotkin (voir chapitre 1) une notion de dérivée partielle. Soit $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$ une fonction booléenne. Alors on notera :

$$D_{x_i}(f)(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_m) = f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_m) - f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_m).$$

6.4.3 Un algorithme naturel

Soit une fonction quadratique $Q : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$. Alors, Q peut s'écrire de la manière suivante :

$$Q(x_1,\ldots,x_m) = Lin(x_1,\ldots,x_m) + Quad(x_1,\ldots,x_m),$$

où $Lin(x_1,\ldots,x_m)$ est la partie purement linéaire de Q et $Quad(x_1,\ldots,x_m)$ est la partie purement quaratique de Q. D'autre part, la forme quadratique $Quad$

peut s'écrire

$$Quad(x_1,\ldots,x_m) = x_1 L_1(x_2,\ldots,x_m) + x_2 L_2(x_3,\ldots,x_m) + \ldots + x_{m-1} L_{m-1}(x_m),$$

où les L_i, $i \in [1,\ldots,m-1]$ sont des fonctions linéaires.

Notons maintenant

$$Quad_i(x_1,\ldots,x_i) = Quad(x_1,\ldots,x_m) - \sum_{j=1}^{i-1} x_j L_j(x_{j+1},\ldots,x_m),$$

alors par construction on a

$$D_{x_i}(Quad_i)(x_{i+1},\ldots,x_m) = L_i(x_{i+1},\ldots,x_m).$$

Maintenant, posons

$$Lin(x_1,\ldots,x_m) = c_0 + c_1 x_1 + \ldots + c_m x_m,$$

et

$$Q_i(x_1,\ldots,x_i) = Q(x_1,\ldots,x_m) - \sum_{j=1}^{i} x_j L_j(x_{j+1},\ldots,x_m),$$

alors

$$D_{x_i}(Q_i)(x_{i+1},\ldots,x_m) = c_i + L_i(x_{i+1},\ldots,x_m). \tag{6.5}$$

On notera par la suite

$$L_i(x_{i+1},\ldots,x_m) = c_{i+1}^{(i)} x_{i+1} + \ldots + c_m^{(i)} x_m.$$

Remarque 16. *On reprendra dans la suite la métrique M que l'on a défini au début de ce chapitre. On remarque que dans l'expression de $D_{x_i}(Q_i)(x_{i+1},\ldots,x_m)$, le terme c_i qui peut être égale à 1 ou 0 apparaît, mais il sera pour nous d'aucune importance puisque du point de vue de notre métrique M, ce terme est sans influence.*

Lemme 18. *Soit f une fonction construite à partir d'une fonction quadratique Q dont on a perturbé par un processus aléatoire, une fraction $\frac{1}{2} - \epsilon$ de ses entrées (canal binaire symétrique). Alors on a*

$$\Delta(D_{x_i}(f), D_{x_i}(Q)) \leq \frac{1}{2} - 2\epsilon^2, \quad i \in [1,\ldots,m].$$

Preuve : Par hypothèse on sait que $\Delta(f,Q) \leq \frac{1}{2} - \epsilon$. Cela peut encore s'écrire

$$\underset{x \in \mathbb{F}_2^m}{\mathbf{Prob}}[f(x) = Q(x)] \geq \frac{1}{2} + \epsilon.$$

Notons $x = (x_1, \ldots, x_m)$, alors l'événement $\{D_{x_i}(f)(x) = D_{x_i}(Q)(x)\}$ est égal à l'union des deux événements :

$$\{\{f(x_i = 1) = Q(x_i = 1)\} \cap \{f(x_i = 0) = Q(x_i = 0)\}\}$$

$$\cup$$

$$\{\{f(x_i = 1) \neq Q(x_i = 1)\} \cap \{f(x_i = 0) \neq Q(x_i = 0)\}\}.$$

Comme, par hypothèse, on est dans les conditions du canal binaire symétrique, les événements $\{\{f(x_i = 1) = Q(x_i = 1)\}$ et $\{f(x_i = 0) = Q(x_i = 0)\}$ sont indépendants et comme la répartition des erreurs est uniformément distribuée, on a

$$\underset{x \in \mathbb{F}_2^m}{\mathbf{Prob}}[f(x) = Q(x)] = \underset{x \in \mathbb{F}_2^m, x_i = y}{\mathbf{Prob}}[f(x) = Q(x)],$$

où $y = 0$ ou 1. Le raisonnement est identique pour les deux événements $\{\{f(x_i = 1) \neq Q(x_i = 1)\}$ et $\{f(x_i = 0) \neq Q(x_i = 0)\}$. On en déduit donc que

$$\underset{x \in \mathbb{F}_2^m}{\mathbf{Prob}}[D_{x_i}(f)(x) = D_{x_i}(Q)(x)] \geq \left(\frac{1}{2} + \epsilon\right)^2 + \left(\frac{1}{2} - \epsilon\right)^2,$$

et finalement

$$\Delta(D_{x_i}(f), D_{x_i}(Q)) \leq \frac{1}{2} - 2\epsilon^2, \quad i \in [1, \ldots, m].$$

□

Lemme 19. *Reprenons les notations précédentes. Soit f une fonction construite à partir d'une fonction quadratique Q dont on a perturbé par un processus aléatoire, une fraction $\frac{1}{2} - \epsilon$ de ses entrées (canal binaire symétrique). Alors on a*

$$M(D_{x_i}(f - Q_i), L_i(x_{i+1}, \ldots, x_m)) \leq \frac{1}{2} - 2\epsilon^2, \quad i \in [1, \ldots, m].$$

Preuve : C'est la conséquence immédiate de la formule 6.5 et du lemme 18. □

Remarque 17. *Dans le canal adversaire, le lemme précédent n'est plus vrai, on peut rencontrer des fonctions f telles que*

$$\Delta(f, RM[2,m]) = \frac{1}{2} - \epsilon,$$

et

$$M(D_{x_i}(f), RM[1,m]) > \frac{1}{2} - 2\epsilon^2.$$

En particulier, cette constatation rend le problème du décodage particulièrement difficile dans le canal adversaire.

L'algorithme que l'on va présenter, fonctionne en deux grandes étapes. La première grande étape consistera à déterminer la partie purement quadratique des meilleurs approximations quadratiques de la fonction f que l'on considère. La deuxième étape consistera à déterminer la partie linéaire des meilleurs approximations quadratiques de cette fonction f. Tout d'abord, rappelons quelques résultats qui vont nous permettre de justifier la complexité de cet algorithme.

On va résumer les travaux récents de Tor Helleseth, Torleiv Klove, et Vladimir I. Levenshtein [28]. Ces travaux permettent d'affirmer que la probabilité d'obtenir plus d'une solution Q satisfaisant $\Delta(f,Q) \leq \frac{1}{2} - \epsilon$ tend vers 0 lorsque le nombre de variables m tend vers l'infini.

Notons $B_C(t)$ l'ensemble des erreurs corrigibles de poids t pour un code linéaire C de paramètres $[n,k,d]$, c'est à dire l'ensemble des erreurs E pour lesquelles le seul mot a du code C considéré qui vérifie $\Delta(a,c+E) \leq t/n$ pour tout $c \in C$ fixé est c. Notons alors $R_C(t) = |B_C(t)|/\binom{n}{t}$, $t \in [0,\ldots,n]$ la fraction d'erreurs corrigibles parmi toutes les erreurs. Le code de Reed-Muller d'ordre r a pour paramètres $[n,k = 1+m+\ldots+\binom{n}{r},d = 2^{m-r}]$. Le poids t de notre erreur est noté $n(1/2-\epsilon)$. On a alors (cf. théorème 28 chapitre 6 [28]) :

$$1 - R_C\left(\lfloor n(1/2-\epsilon)\rfloor\right) \lesssim \frac{1}{\epsilon}\sqrt{\frac{2n(2^r-1)}{2^r\pi}}e^{-\frac{n\epsilon^2}{2^{r+1}}}.$$

En particulier pour le Reed-Muller d'ordre 2, on a la formule

$$1 - R_C\left(\lfloor n(1/2-\epsilon)\rfloor\right) \lesssim \frac{1}{\epsilon}\sqrt{\frac{3n}{2\pi}}e^{-\frac{n\epsilon^2}{8}}.$$

Tout ceci montre que, si l'on considère un nombre suffisant de variables m, avec ϵ fixé, tel que, $n = \mathcal{O}(\epsilon^{-2})$ à l'issue de notre décodage, il ne restera qu'une seule solution à notre problème. Ceci implique aussi que si l'on applique notre algorithme adaptatif de reconstruction à la fonction $D_{x_i}(f-Q_i)$, en prenant soin de choisir les paramètres en fonction du canal binaire symétrique, et un nombre de variables m tel que $n = \mathcal{O}(\epsilon^{-4})$), alors il ne résultera qu'une seule fonction affine quelque soit l'entier $i \in [1,\ldots,m]$ choisi.

Détermination de la partie quadratique

Rappelons que l'on a appelé *Quad* la partie purement quadratique d'une fonction quadratique Q qui vérifie $\Delta(f,Q) \leq \frac{1}{2} - \epsilon$. Rappelons que l'on a appelé *"Reconstruct-linear-fast"* notre algorithme adaptatif rapide de reconstruction pour les codes de Reed-Muller d'ordre un de la figure 6.9. Appliquons le à la fonction $D_{x_1}(f)$. On obtient alors une fonction linéaire

$$L_1(x_2,\ldots,x_m) = c_2^{(1)}x_2 + \ldots + c_m^{(1)}x_m.$$

Maintenant on peut itérer le procédé avec la fonction $D_{x_2}(f - Q_1)$, on obtiendra alors

$$L_2(x_3, \ldots, x_m) = c_2^{(2)} x_2 + \ldots + c_m^{(2)} x_m.$$

Puis à la dernière étape, on appliquera le programme *"Reconstruct-linear-fast"* à la fonction

$$D_{x_{m-1}}(f - Q_{m-1}).$$

On a ainsi déterminé la partie purement quadratique d'une fonction quadratique Q qui vérifie $\Delta(f,Q) \leq \frac{1}{2} - \epsilon$. On a représenté qui effectuera cette reconstruction à la la figure 6.10.

```
Quad-part(f,ε,m,c,u)

  M := {}; (c'est une liste de fonctions linéaires).
  Q := {}; (c'est une liste de fonctions quadratiques).
  L := {0}; (c'est une liste de fonctions quadratiques).
  T := {}; (c'est une liste de fonctions quadratiques).

 pour i de 1 à m − 1 faire
    {
      T := {};
      pour v de 1 à card(L) faire
        {
          M := Reconstruct-linear-fast(D_{x_i}(f − L[v]),2ε²,m − i,c,u);
          Q := [.];

          pour j de 1 à card(M) faire
            {
              Q[j] := L[v] + x_i · M[j];
            }

          T := T ∪ Q;
        }

      L := T;
    }
 retourner L;
```

FIG. 6.10 – *Reconstruction de la partie quadratique*

Détermination de la partie linéaire

On suppose maintenant que l'on a déjà reconstruit les parties purement quadratiques *Quad* des fonctions quadratiques Q qui satisfont $\Delta(f,Q) \leq \frac{1}{2} - \epsilon$. Alors on obtient aisément la partie purement linéaire en appliquant le programme *"Reconstruct-linear-fast"* sur les fonctions

$$f(x_1,\ldots,x_m) - Quad(x_1,\ldots,x_m).$$

L'algorithme finale de reconstruction est représenté à la figure 6.11

```
Reconstruct-quadratic(f,ε,m,u,c)

   Q := {}; (c'est une liste de fonctions purement quadratiques).
   L := {};  (c'est une liste de fonctions quadratiques).
   M := {};  (c'est une liste de fonctions linéaires).
   T := {};  (c'est une liste de fonctions quadratiques).

   Q := Quad-part(f,ε,m,c,u);

  pour i de 1 à card(Q) faire
     {
       M := Reconstruct-linear(f − Q[i],ε,m,c,u);
       T := [.]

       pour j de 1 à card(M) faire
        {
          T[j] := Q[i] + M[j];
        }

       L = L ∪ T;
     }

  retourner L;
```

FIG. 6.11 – *Algorithme de reconstruction quadratique*

Théorème 38. *Soient* $f : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$ *tel que* $\Delta(f,RM[2,m]) \leq \frac{1}{2} - \epsilon$. *Supposons que la fonction* f *est équivalente à une fonction quadratique dont on aurait perturbé aléatoirement une fraction d'au plus* $\frac{1}{2} - \epsilon$ *de ses images par un processus aléatoire. Supposons de plus que le nombre de variables soit si grand, qu'il n'y*

ait qu'une seule fonction quadratique Q qui vérifie $\Delta(f,Q) \leq 1/2 - \epsilon$ et tel qu'il n'y ait qu'une seule fonction affine C qui vérifie $\Delta(D_{x_i}(f),C) \leq 1/2 - 2\epsilon^2$ pour tout $i \in [1,...,m]$, c'est-à-dire $2^m = \mathcal{O}(\epsilon^{-4})$. Alors le problème de reconstruction à clairs choisis dans le canal binaire symétrique peut être résolu avec une complexité de l'ordre de

$$\mathcal{O}\left(\frac{((u+\log(2m^2))m^3}{\epsilon^4}\right)$$

opérations binaires et avec une probabilité d'échec inférieure à 2^{-u} en utilisant notre algorithme adaptatif de reconstruction quadratique 6.11.

Preuve : Le fait qu'il n'y ait qu'une seule fonction linéaire C telle que $\Delta(D_{x_i}(f),C) \leq \frac{1}{2} - \epsilon$ pour tout $i \in [1,...,m]$ est justifié par le théorème 28 du chapitre 6. L'évaluation d'une fonction quadratique à m variables coûte $\mathcal{O}(m^2)$ Donc d'après le théorème 36 du chapitre 6 le coût de la reconstruction de la partie quadratique est de l'ordre de

$$\mathcal{O}\left(\frac{(u+\log(2m^2))m^3}{\epsilon^4}\right),$$

avec une probabilité d'échec inférieure à 2^{-u}. Une fois la partie quadratique reconstruite, il reste à déterminer le coût de la partie linéaire. Une fonction quadratique comportant m variables peut être évaluée en $\mathcal{O}(m^2)$ opérations binaires, donc la reconstruction de la partie affine a un coût de l'ordre de

$$\mathcal{O}\left(\frac{(u+\log(2m))m^3}{\epsilon^2}\right),$$

avec un probabilité d'échec inférieure à 2^{-u}. Donc finalement ce problème de reconstruction peut être résolu en

$$\mathcal{O}\left(\frac{(u+\log(2m^2))m^3}{\epsilon^4}\right),$$

avec une probabilité d'échec inférieure à 2^{-u}. □

Remarque 18. *La fonction* "Reconstruct-quadratic" *de la figure 6.9 prend en argument les paramètres ϵ,u,c, de même que la fonction* "Reconstruct-linear-fast" *de la figure 6.9. Ceci vient du fait que pour effectuer un échantillonnage, il faut se donner une probabilité (ici: $1/2+\epsilon$), l'écart-type de l'erreur (ϵ/c avec $c > 1$), et une probabilité de succès. La taille de l'échantillon dépend alors de ces trois quantités*

6.4.4 Quelques résultats expérimentaux

On a expérimentalement testé cet algorithme de reconstruction quadratique dans trois contextes différents. Dans le premier contexte on s'est placé dans les

conditions du canal binaire symétrique, avec un grand nombre de variables, par exemple 64. Dans le deuxième contexte on considère toujours le canal binaire symétrique, mais avec peu de variables, c'est-à-dire moins de 11 variables. Enfin dans le troisième contexte on a considéré le canal adversaire.

Un grand nombre de variables dans le canal binaire symétrique

On a construit une fonction :

$$f : \mathbb{F}_2^{64} \longrightarrow \mathbb{F}_2$$

qui coïncide avec la fonction quadratique $Q : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$, telle que

$$Q(x_1, \ldots, x_{64}) = x_7 + x_{15} + x_3x_{13} + x_{11}x_{19} + x_{12}x_{63} + x_{25}x_{44} + x_{33}x_{40} + x_{10}x_{62}$$

sur une fraction $\frac{1}{2}+\epsilon$ de ces entrées. Les images de cette fonction ont été perturbé par un processus aléatoire. On a implémenté cette fonction d'une telle manière que l'on peut faire varier le paramètre ϵ. A chaque fois, pour les paramètres que l'on a considérés, la taille de la liste au cours de l'algorithme est toujours restée égale à un. Un PC à 1.5 GHz a été utilisé. On a donc représenté le temps de calcul en secondes en fonction de ϵ :

ϵ	0.45	0.4	0.35	0.3	0.25	0.2	0.1
temps	7	16	97	118	565	928	52624

Un petit nombre de variables dans le canal binaire symétrique

On a construit une fonction :

$$f : \mathbb{F}_2^{10} \longrightarrow \mathbb{F}_2$$

qui coïncide avec la fonction quadratique $Q : \mathbb{F}_2^m \longrightarrow \mathbb{F}_2$, telle que

$$Q(x_1, \ldots, x_{10}) = x_7 + x_1 + x_3x_1 + x_{10}x_9 + x_2x_6 + x_4x_{10}$$

sur une fraction $\frac{1}{2} + \epsilon$ de ces entrées. Les images de cette fonction ont été perturbées par un processus aléatoire. On a implémenté cette fonction d'une telle manière que l'on peut faire varier le paramètre ϵ. Notre algorithme de reconstruction quadratique ne nécessite plus maintenant d'échantillonnage, pour obtenir les approximations de $D_{x_i}(f)$ on a simplement utilisé des transformées de Fourier rapides. La complexité de notre algorithme dans ce cas est de l'ordre de

$$\mathcal{O}(m \cdot 2^{2m}).$$

Dans ce cas l'algorithme que l'on propose n'est efficace que pour des valeurs de ϵ supérieures à 0.2, ce qui correspond à moins de 30 pourcent de bruit. On a donc

représenté le temps en secondes en fonction du nombre de décodages que l'on effectue pour $\epsilon = 0.2$:

Nombre de décodages	40	50	80	100	200	400	1000
temps	0	1	2	2	4	8	16

Pour des valeurs de ϵ plus petites, la liste des candidats s'accroit de manière dramatique lors du décodage. Notre algorithme est alors sans intérêt, puisque le temps de calcul est trop long. Nous allons donner un exemple pour $\epsilon = 0.17$. On a représenté la taille de la liste au cours des $m = 10$ étapes de notre algorithme:

Etape de l'algo.	Nombre de candidats
1	1
2	2
3	6
4	42
5	168
6	504
7	1512
8	6048
9	12096
10	15

Ce calcul a mis 315 secondes. Donc ce type d'algorithme reste assez limité pour ces paramètres (paramètres qui sont utilisés dans les télécommunications).

Le canal adversaire

Dans ce type de canal, on a construit une fonction f de la manière suivante. On considère la fonction quadratique

$$Q(x_1,\ldots,x_{10}) = x_7 + x_{15} + x_3x_{13} + x_{11}x_{19} + x_{12}x_{63} + x_{25}x_{44} + x_{33}x_{40} + x_{10}x_{62}.$$

Cette fonction prend en entrée des éléments de $\mathbb{F}_2^{64}$ qui sont en bijection avec l'ensemble des entiers compris entre 0 et $2^{64}-1$. Alors on définit f comme:

$$f(x) = \begin{cases} Q(x) & \text{si } x \leq (2^{64}-1) \times (\frac{1}{2}+\epsilon) \\ Q(x)+1 & \text{si } x > (2^{64}-1) \times (\frac{1}{2}+\epsilon). \end{cases}$$

Par construction, on a $\Delta(f,Q) \leq \frac{1}{2} - \epsilon$. Cependant notre algorithme n'a pas pu reconstruire la forme quadratique Q pour des valeurs de ϵ plus petites que 0.25. Le problème rencontré est le suivant: pour plusieurs valeurs de $i \in [1,...,m]$ on a obtenu

$$M(D_{x_i}(f),RM[1,m]) \gg \frac{1}{2} - \epsilon.$$

Donc voici un phénomène qui peut faire échouer notre algorithme. Un deuxième phénomène est celui rencontré dans le cas d'un petit nombre de variables, la liste peut s'accroitre de telle sorte que le temps de calcul ne soit plus raisonnable. Par exemple, on peut construire la fonction $f : \mathbb{F}_2^{2m} \longrightarrow \mathbb{F}_2$ définie par

$$f(x_1,\ldots,x_{2m}) = x_1x_2 + x_3x_4 + x_5x_6 + \ldots + x_{2m-1}x_{2m}.$$

Maintenant on veut trouver toutes les fonctions quadratiques $q : \mathbb{F}_2^{2m} \longrightarrow \mathbb{F}_2$ telles que $\Delta(f,q) \leq \frac{1}{2} - \epsilon$ avec $\epsilon = 0.25$. Il est facile de vérifier que

$$\Delta(x_1x_2,x_1) = \frac{1}{2} - \epsilon$$

donc si

$$q(x_1,\ldots,x_{2m}) = x_1 + x_3x_4 + x_5x_6 + \ldots + x_{2m-1}x_{2m},$$

on a bien $\Delta(f,q) \leq \frac{1}{2} - \epsilon$. On peut construire de cette manière au moins $2m$ fonctions quadratiques q telles que :

$$q(x_1,\ldots,x_{2m}) = x_1x_2 + \ldots + x_{i-1}x_{i-2} + x_i + x_{i+1}x_{i+2} + \ldots + x_{2m-1}x_{2m}.$$

Donc, à $\epsilon = 0.25$ fixé on peut construire une fonction f telle que $\Delta(f,RM[2,m]) = \frac{1}{2} - \epsilon$, et telle qu'il existe une liste de taille au moins $2m$, de fonctions quadratiques qui sont à une distance inférieure à $\frac{1}{2} - \epsilon$. Dans ce cas, il n'est pas possible d'avoir un algorithme qui ferait du décodage par liste avec une complexité polynomiale en $1/\epsilon$.

Conclusion

Les conditions d'utilisation d'un tel algorithme sont beaucoup plus limitées que l'algorithme de reconstruction linéaire. Cependant, si le côté adversaire n'est pas trop redoutable, et si le nombre de variable est suffisant, cet algorithme peut donner de très bon résultats.

6.5 Algorithme de décodage non adaptatif

Cette section résulte d'un travail en collaboration avec Gregory Kabatianski. On a jusqu'ici étudié des algorithmes adaptatifs pour répondre au problème de reconstruction. Dans le cas de la recherche des meilleurs approximations affines d'une fonction f, cela signifie que l'on pose des questions à la fonction f (par exemple ses valeurs en certains points) et en fonction des réponses on va prendre des décisions. Ce type d'algorithme implique que l'on travaille à clairs choisis. Mais on aimerait résoudre ce problème à clairs connus. C'est-à-dire que l'on se donne un échantillon aléatoire d'entrées-sorties de la fonction f, et en fonction de cet échantillon on aimerait retrouver les fonctions affines ayant la plus grande coïncidence avec f sur l'ensemble de ses entrées.

6.5.1 Une idée d'algorithme

On considère toujours une fonction booléenne $f : \mathbb{F}_2^m \to \mathbb{F}_2$ telle que

$$\Delta(f,RM[1,m]) \leq \frac{1}{2} - \epsilon.$$

Notre idée est naturelle, on aimerait utiliser les mêmes algorithmes que dans la section 3 du chapitre 6. Simplement, on aimerait borner la taille minimum de l'échantillon nécessaire au décodage. Cette idée fut d'abord développée par T. Johansson et F. Jönsson [7] dans le canal binaire symétrique. On va, nous aussi analyser le problème dans le canal binaire symétrique. L'idée de T. Johansson et F. Jönsson [7] est de prendre autant d'entrées-sorties qu'il est nécessaire pour retrouver les entrées choisies d'un algorithme de reconstruction adaptatif tel que ceux qui figure dans le chapitre 6. T. Johansson et F. Jönsson ont montré que la complexité de cette méthode est exponentielle en $1/\epsilon$. On a montré dans le chapitre 6, que pour les paramètres, que l'on considère (ϵ), il faut connaître au moins $1/\epsilon^2$ entrée-sorties de la forme

$$(\overline{r},\overline{s}) \in \mathbb{F}_2^i \times \mathbb{F}_2^{m-i},$$

avec $\overline{s}$ fixé, pour pouvoir effectuer la i-ième étape de l'algorithme adaptatif de reconstruction linéaire du chapitre 6. Notre idée est de prendre un échantillon aléatoire de taille $poly(1/\epsilon^2)$, et de trouver un isomorphisme d'espace vectoriel ϕ tel qu'il existe au moins ϵ^{-2} éléments $(\overline{r}_t,\overline{s})$ de $\mathbb{F}_2^m$ satisfaisant

$$\phi(x_t) = (\overline{r}_t,\overline{s}), \quad t \in [1,...,\epsilon^{-2}].$$

L'algorithme suggéré fonctionne alors de la manière suivante :

On garde les m étapes de l'algorithme 6.7. Supposons que l'on est à la i-ième étape. On tire alors aléatoirement $poly(1/\epsilon^2)$ éléments de $\mathbb{F}_2^m$, puis on détermine

un isomorphisme ϕ tel que pour au moins ϵ^{-2} éléments parmi ceux que l'on a tiré, on ait la relation :

$$\phi(x_t) = (\overline{r}_t, \overline{s}), \quad t \in [1,\ldots,\epsilon^{-2}]$$

pour $\overline{s}$ fixé, on peut alors déterminer la meilleure approximation affine de la fonction $f \circ \phi^{-1}(x, \overline{s})$ de la même manière que pour l'algorithme 6.7. Par inversion de l'isomorphisme ϕ on obtiendra alors une relation linéaire entre les coefficients de la meilleure approximation affine de f. En répétant cette opération $\mathcal{O}(m)$ fois, on obtient un système d'équation de rang 1 à résoudre et on en déduit la meilleure approximation affine de f.

6.5.2 Une impasse

On va reprendre les notations utilisés dans le chapitre 6. On va montrer dans cette section que de manière générale, ces transformations linéaires bijectives ne permettent pas d'obtenir un algorithme polynômial en le logarithme de la longueur du code de Reed-Muller, c'est-à-dire m avec $n = 2^m$. On va par exemple montrer que si $\log_2(\epsilon^{-2}) = m/2$, alors l'algorithme suggéré est exponentiel en m. Rappelons qu'à la i-ième étape de notre algorithme adaptatif de reconstruction linéaire du chapitre 6, il faut évaluer la quantité

$$M(f(x_1, \ldots, x_i, \overline{s}), c_1 x_1 + \ldots + c_i x_i),\ \overline{s} \in \mathbb{F}_2^{m-i}.$$

Pour un paramètre ϵ fixé il faut donc évaluer f en $\mathcal{O}(poly(m))$ entrées de la forme $(r_t, \overline{s})$, $t \in [1, \ldots, 1/\epsilon^2]$, où $\overline{s}$ est fixé. Ces éléments forment un sous espace de dimension i. Notons A_i l'ensemble des sous-espaces vectoriels de $\mathbb{F}_2^m$ de dimension i. Notons X l'échantillon d'éléments de $\mathbb{F}_2^m$ que l'on va tirer aléatoirement. On supposera que

$$x = |X| = poly(m) > m,$$

Alors on va s'intéresser à la quantité

$$\tau_i(X) = \max |X \cap A_i|.$$

On peut majorer la quantité $|A_i|$:

$$|A_i| = \frac{(2^m - 1)(2^m - 2)\ldots(2^m - 2^{m-i+1})}{(2^i - 1)(2^{i-1} - 1)\ldots(2 - 1)} \leq \frac{(2^m)^i}{2^l \times 2^{l-1} \ldots \times 2^2 \times 1}$$

d'où

$$|A_i| \leq 2^{i(m - \frac{i}{2})}.$$

Lemme 20. *Soit L un sous espace-vectoriel de $\mathbb{F}_2^m$ de dimension i. Alors*

$$\underset{a \in \mathbb{F}_2^m}{\mathbf{Prob}} [a \in L] = \frac{2^i}{2^m} = 2^{i-m}.$$

Lemme 21. *Soit L un sous espace-vectoriel de $\mathbb{F}_2^m$ de dimension i. Notons $\alpha = 2^{i-m}$ et T un entier. Alors on a :*

$$\underset{X\in(\mathbb{F}_2^m)^x}{\mathbf{Prob}}\left[|X\cap L|>T\right]\leq\left(\frac{x\alpha e}{T}\right)^T.$$

Preuve : On a

$$\underset{X\in(\mathbb{F}_2^m)^x}{\mathbf{Prob}}\left[|X\cap L|>T\right]=\sum_{j=T}^{x}\binom{x}{i}\alpha^j(1-\alpha)^{x-j}\leq x\binom{x}{T}\alpha^T$$

d'où le lemme. □

Corollaire 1. *Posons $x = m^u$, $T = m^{1+\mu}$ et $i = m/2$. Avec les notations précédentes*

$$\underset{X\in(\mathbb{F}_2^m)^x}{\mathbf{Prob}}\left[\tau_i(X)>T\right]\leq 2^{\frac{m^2}{2}\left(\frac{3}{4}-m^\mu\right)}\left(em^{u-1-\mu}\right)^{m^{1+\mu}}$$

Preuve : Si L désigne un sous espace de dimension i de $\mathbb{F}_2^m$, on a

$$\underset{X\in(\mathbb{F}_2^m)^x}{\mathbf{Prob}}\left[\tau_i(X)>T\right]\leq\underset{X\in(\mathbb{F}_2^m)^x}{\mathbf{Prob}}\left[|X\cap L|>T\right]\times\underset{a\in\mathbb{F}_2^m}{\mathbf{Prob}}\left[a\in L\right],$$

donc d'après le lemme précédent

$$\underset{X\in(\mathbb{F}_2^m)^x}{\mathbf{Prob}}\left[\tau_i(X)>T\right]\leq 2^{i\left(m-\frac{i}{2}\right)}\times\left(\frac{x\alpha e}{T}\right)^T.$$

□

On vient donc dans le corollaire précédent d'exhiber un exemple où

$$\log\left(\frac{1}{\underset{X\in(\mathbb{F}_2^m)^x}{\mathbf{Prob}}\left[\tau_i(X)>T\right]}\right)=\mathcal{O}\left(m^{2+\mu}\right),$$

donc la méthode qui consisterait à effectuer des transformations bijectives sur les entrées de la fonction f que l'on considère ne permet pas d'obtenir un algorithme polynômial en le nombre de variables m pour tout ϵ.

6.6 Application à la cryptanalyse du DES

Cette partie a été réalisée en collaboration avec Eric Filiol. Dans cette partie, nous allons utiliser les algorithmes de reconstruction de polynômes multivariés que l'on a développés jusqu'à présent pour tenter de faire une analyse cryptographique du DES. On a repris les Travaux de Matsui [35]. Dans le chapitre 2 on a complètement décrit le DES. Dans le chapitre 3, on a donné une description des attaques de Matsui, et notamment sa recherche d'expressions linéaires.

6.6.1 Quelques notations

Rappelons que le DES peut être assimilé à une fonction du type

$$Des : \mathbb{F}_2^{128} \longrightarrow \mathbb{F}_2^{64}.$$

C'est-à-dire que c'est une fonction qui prend en entrée 64 premiers bits que l'on appelle le clair qui se décompose elle-même en 2 blocs de 32 bits notés habituellement (L_0,R_0), où L signifie "left" et R "right", le 0 correspond au tour numéro 0. Le bloc des 64 autres bits qui suivent correspondent à la clef et est noté K. La clef du DES ne comporte que 56 bits, en fait parmi les 64 bits de clefs qui sont pris en entrée, 8 bits n'interviennent pas. La sortie est composée de 64 bits qui se décompose en 2 blocs de 32 bits notés habituellement (L_t,R_t) s'il s'agit du DES à t tours. Ce couple est généralement appelé chiffré. Pour résumer, on a pour le DES à t tours

$$DES_t(L_0,R_0,K) = (L_t,R_t).$$

On conservera les notations utilisées précédemment lorsque l'on parlera de décodage et de distance.

6.6.2 Les objectifs

On aimerait obtenir des relations linéaires entre le clair, le chiffré et la clef du type

$$\sum_{i=1}^{32} \alpha_i L_0[i] + \sum_{i=1}^{32} \beta_i R_0[i] + \sum_{i=1}^{64} k_i K[i] = \sum_{i=1}^{32} \lambda_i L_t[i] + \sum_{i=1}^{32} \mu_i R_t[i],$$

avec $\alpha_i,\beta_i,k_i,\lambda_i,\mu_i \in \{0,1\}$ et qui sont réalisées avec une probabilité $\delta = \frac{1}{2} + \epsilon$. Notre idée est donc d'utiliser notre algorithme de reconstruction linéaire, voire quadratique, afin de trouver plus d'équations que celle que l'on connaît déjà qui sont dues à Matsui. On espère entre autres que les biais que l'on va obtenir sont meilleurs que les biais des équations de Matsui. Par construction, le couple (L_t,R_t) dépend du triplet (L_0,R_0,K) donc on ne pourra pas appliquer directement notre

algorithme de reconstruction linéaire à la fonction booléenne $g : \mathbb{F}_2^{191} \longrightarrow \mathbb{F}_2$ définie par

$$g(R_0,L_0,K,R_t,L_t[1],\ldots,L_t[l-1],L_t[l+1],\ldots,L_t[32]) = L_t[l],\ l \in [1,...,32].$$

Ceci car notre algorithme est un algorithme à clairs choisis, de plus on a montré dans la section précédente que des transformations simple ne permettaient pas de transformer notre algorithme en algorithme à clairs connus. Il faut donc se donner un ensemble de λ_i et μ_i fixés à l'avance. On appliquera ensuite nos algorithmes à la fonction $f : \mathbb{F}_2^{128} \longrightarrow \mathbb{F}_2$ définie par

$$f(R_0,L_0,K) = \sum_{i=1}^{32} \lambda_i L_t[i] + \sum_{i=1}^{32} \mu_i R_t[i],$$

avec $\lambda_i,\mu_i \in \{0,1\}$.

6.6.3 Notre stratégie

On rappelle qu'obtenir une équation linéaire du type précédent sur 16 tours permet d'obtenir immédiatement une combinaison linéaire de certain bits de clef. Si l'on obtient suffisamment d'équations on peut en déduire plusieurs combinaisons linéaires de bits de clef et on peut espérer ainsi obtenir la totalité des 56 bits de clef. Comme nous l'avons dit, nous n'avons pas de moyens algorithmiques qui nous permettraient de choisir la meilleure combinaison linéaire

$$\sum_{i=1}^{32} \lambda_i L_t[i] + \sum_{i=1}^{32} \mu_i R_t[i]$$

pour t tours. On sait simplement grâce aux travaux de Matsui calculer cette meilleure combinaison linéaire au premier tour. On va donc pour les premiers tours utiliser les mêmes combinaisons linéaires que Matsui. Puis si l'on obtient différentes approximations linéaires, on pourra alors les considérer comme des combinaisons linéaires potentielles pour le DES à plusieurs tours. On a fait ceci car on a pas vraiment de méthodes qui permettent de trouver les meilleurs sorties pour le DES à t tours. Enfin, on pourra éventuellement utiliser des approximations quadratiques dans l'espoir d'obtenir des biais statistiques plus importants que ceux obtenus par Matsui. On a représenté dans les tableaux suivants les approximations linéaires que l'on a obtenues, ainsi que diverses informations comme le temps de calcul en secondes. On a, à chaque fois, retrouvé l'équation de Matsui. On a mis le symbole (*) lorsqu'il s'agissait de l'équation de Matsui. Il faut enfin noter, qu'expérimentalement, on a pu obtenir un plus grand nombre d'équations qu'il ne figure dans les tableaux qui suivent, on s'est contenté de présenter l'échantillon des solutions les plus proches. Bien sûr notre algorithme ne permet

pas d'obtenir des équations linéaires directement sur 14, 15 ou 16 tours car le nombre d'opérations à effectuer serait trop grand ($> 2^{65}$ opérations binaires). On essaie donc de trouver des relations linéaires sur plusieurs tours que l'on puisse chaîner, du type :

$$\sum_{i=1}^{32} \alpha_i L_0[i] + \sum_{i=1}^{32} \beta_i R_0[i] + \sum_{i=1}^{64} k_i K[i] = \sum_{i=1}^{32} \lambda_i L_t[i] + \sum_{i=1}^{32} \mu_i R_t[i],$$

pour t tours et pour $t' - t$ tours :

$$\sum_{i=1}^{32} \alpha_i L_t[i] + \sum_{i=1}^{32} \beta_i R_t[i] + \sum_{i=1}^{64} k'_i K[i] = \sum_{i=1}^{32} \lambda'_i L_{t'}[i] + \sum_{i=1}^{32} \mu'_i R_{t'}[i].$$

Ces deux équations donnent donc l'équation sur t' tours :

$$\sum_{i=1}^{32} \alpha_i L_0[i] + \sum_{i=1}^{32} \beta_i R_0[i] + \sum_{i=1}^{64} (k_i + k'_i) K[i] = \sum_{i=1}^{32} \lambda_i L_{t'}[i] + \sum_{i=1}^{32} \mu_i R_{t'}[i].$$

Évidemment cette opération n'est pas sans conséquences sur les biais :

Lemme 22. *(piling-up lemma) [34]. Soient X_i $(1 \leq i \leq s)$ des variables aléatoires indépendantes égales à 0 avec probabilité p_i et égalent à 1 avec probabilité $1 - p_i$. Alors la probabilité que $X_1 + \ldots + X_s = 0 \mod 2$ est*

$$\frac{1}{2} + 2^{s-1} \prod_{i=1}^{s} \left(p_i - \frac{1}{2} \right).$$

Dans la pratique on remarquera que l'hypothèse "variables indépendantes" est un peu exagérée, mais cette méthode donne tout de même une idée assez réaliste des biais que l'on peut trouver en chaînant de telles équations linéaires. Dans la suite comme les bits de clef sont pour nous des variables, on a a représenté l'approximation affine $\sum_{i=1}^{32} \alpha_i L_0[i] + \sum_{i=1}^{32} \beta_i R_0[i] + \sum_{i=1}^{64} k_i K[i]$ par une combinaison linéaire des x_i, $i \in [1,\ldots,128]$. De même on a représenté $\sum_{i=1}^{32} \lambda_i L_t[i] + \sum_{i=1}^{32} \mu_i R_t[i]$ par une combinaison linéaire des x_i, $i \in [1,\ldots,64]$.

Le DES à 1 tour

Combinaison linéaire de sortie : $x_3 + x_8 + x_{14} + x_{25}$ (sortie de Matsui).

ϵ	Equations obtenues	temps
0.187	$x_{16} + x_{20} + x_{35} + x_{40} + x_{46} + x_{57} + x_{86} + x_{101}$	0
0.311	(*) $x_{17} + x_{35} + x_{40} + x_{46} + x_{57} + x_{92}$	

Combinaison linéaire de sortie : $x_1 + x_2 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32}$.

ϵ	Equations obtenues	temps
0.1249	$x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{98} + x_{113}$	
0.1252	$x_1 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{81} + x_{98} + x_{113}$	
0.1564	$x_2 + x_3 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{81} + x_{124}$	
0.1874	$x_3 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{81} + x_{98} + x_{124}$	0
0.1248	$x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{113} + x_{124}$	
0.1562	$x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{98} + x_{113} + x_{124}$	
0.1562	$x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{98} + x_{113} + x_{124}$	

Combinaison linéaire de sortie : x_2.

ϵ	Equations obtenues	temps
0,218	$x_4 + x_5 + x_7 + x_8 + x_9 + x_{34} + x_{73} + x_{83} + x_{97} + x_{106} + x_{121}$	0
0.219	$x_4 + x_7 + x_8 + x_{34} + x_{73} + x_{83} + x_{97}$	

Combinaison linéaire de sortie : x_{31}.

ϵ	Equations obtenues	temps
0.187	$x_1 + x_3 + x_4 + x_5 + x_{63} + x_{81} + x_{124}$	0
0.218	$x_1 + x_2 + x_4 + x_5 + x_{32} + x_{63} + x_{74} + x_{81} + x_{98}$	

Combinaison linéaire de sortie : $x_8 + x_{14} + x_{25}$.

ϵ	Equations obtenues	temps
0.124	$x_{17} + x_{40} + x_{46} + x_{57} + x_{92}$	
0.124	$x_{16} + x_{20} + x_{40} + x_{46} + x_{57} + x_{86} + x_{101}$	
0.125	$x_{17} + x_{20} + x_{40} + x_{46} + x_{57} + x_{92} + x_{101}$	
0.093	$x_{16} + x_{17} + x_{19} + x_{40} + x_{46} + x_{57} + x_{68} + x_{86} + x_{92} + x_{103}$	0
0.125	$x_{17} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{68} + x_{92} + x_{118}$	
0.124	$x_{16} + x_{17} + x_{19} + x_{40} + x_{46} + x_{57} + x_{86} + x_{92} + x_{118}$	
0.156	$x_{18} + x_{19} + x_{40} + x_{46} + x_{57} + x_{103} + x_{118}$	
0.156	$x_{18} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{68} + x_{103} + x_{118}$	

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{34} + x_{36} + x_{37} + x_{40} + x_{46} + x_{57} + x_{64}$.

ϵ	Equations obtenues	temps
0.156	$x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{98} + x_{113} + x_{124}$	
0.124	$x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{113} + x_{124}$	
0.187	$x_3 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{81} + x_{98} + x_{124}$	
0.156	$x_2 + x_3 + x_8 + x_{14} + x_{25} + x_{49} + x_{74} + x_{81} + x_{124}$	
0.093	$x_2 + x_3 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{81} + x_{124}$	
0.125	$x_1 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{81} + x_{98} + x_{113}$	0
0.125	$x_4 + x_5 + x_{14} + x_{25} + x_{32} + x_{49} + x_{98} + x_{113}$	
0.093	$x_1 + x_4 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{81} + x_{98}$	
0.093	$x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{81} + x_{98}$	
0.093	$x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{98}$	

Pour le premier tour l'équation de Matsui possède le meilleur biais, et ce biais est sans surprise celui qui figure dans l'article de Matsui [34].

Le DES à 2 tours

Combinaison linéaire de sortie : $x_8 + x_{14} + x_{25}$.

ϵ	Equations obtenues	temps
0.041	$x_5 + x_6 + x_7 + x_9 + x_{14} + x_{25} + x_{50} + x_{66} + x_{73} +$ $x_{83} + x_{95} + x_{106} + x_{121}$	
0.038	$x_1 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{81} + x_{84} + x_{124}$	
0.046	$x_1 + x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{81} + x_{84} + x_{98} + x_{124}$	6
0.031	$x_1 + x_3 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{84} + x_{113} + x_{124}$	
0.039	$x_1 + x_2 + x_3 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74}$ $x_{84} + x_{98} + x_{113} + x_{124}$	

Combinaison linéaire de sortie : $x_{40} + x_{46} + x_{57}$.

ϵ	Equations obtenues	temps
0.124	$x_{17} + x_{40} + x_{46} + x_{57} + x_{92}$	
0.124	$x_{16} + x_{20} + x_{40} + x_{46} + x_{57} + x_{86} + x_{101}$	
0.125	$x_{17} + x_{20} + x_{40} + x_{46} + x_{57} + x_{92} + x_{101}$	
0.125	$x_{17} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{68} + x_{92} + x_{118}$	0
0.124	$x_{16} + x_{17} + x_{19} + x_{40} + x_{46} + x_{57} + x_{86} + x_{92} + x_{118}$	
0.155	$x_{18} + x_{19} + x_{40} + x_{46} + x_{57} + x_{103} + x_{118}$	
0.155	$x_{18} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{68} + x_{103} + x_{118}$	

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{34} + x_{36} + x_{37} + x_{40} + x_{46} + x_{57} + x_{64}$.

ϵ	Equations obtenues	temps
0.117	$x_{35} + x_{40} + x_{46} + x_{57} + x_{66} + x_{73} + x_{90} + x_{92} +$ $x_{105} + x_{107} + x_{116}$	0
0.07	$x_{16} + x_{17} + x_{20} + x_{35} + x_{40} + x_{46} + x_{57} + x_{66} + x_{73} +$ $x_{86} + x_{90} + x_{101} + x_{105} + x_{107} + x_{116}$	

Combinaison linéaire de sortie : $x_{17} + x_{35} + x_{40} + x_{46} + x_{57}$.

ϵ	Equations obtenues	temps
0.01	$x_4 + x_7 + x_8 + x_{12} + x_{15} + x_{16} + x_{33} + x_{34} + x_{35} +$ $x_{40} + x_{46} + x_{57} + x_{73} + x_{74} + x_{83} + x_{90} + x_{91} +$ $x_{92} + x_{97} + x_{103} + x_{107}$	
0.011	$x_4 + x_7 + x_8 + x_{12} + x_{13} + x_{14} + x_{33} + x_{34} + x_{35} +$ $x_{40} + x_{46} + x_{57} + x_{65} + x_{73} + x_{83} + x_{90} + x_{92} +$ $x_{97} + x_{100} + x_{103} + x_{107}$	
0.013	$x_{17} + x_{18} + x_{19} + x_{21} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} +$ $x_{29} + x_{40} + x_{46} + x_{57} + x_{64} + x_{66} + x_{68} + x_{79} + x_{84} +$ $x_{85} + x_{102} + x_{103} + x_{116} + x_{118} + x_{125} + x_{127}$	
0.013	$x_{17} + x_{18} + x_{19} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} + x_{29} + x_{40} +$ $x_{46} + x_{57} + x_{64} + x_{66} + x_{79} + x_{84} + x_{85} + x_{102} + x_{103} +$ $x_{116} + x_{118} + x_{125} + x_{127}$	
0.011	$x_{17} + x_{18} + x_{19} + x_{21} + x_{24} + x_{26} + x_{27} + x_{28} + x_{29} + x_{40} +$ $x_{46} + x_{57} + x_{64} + x_{66} + x_{68} + x_{79} + x_{84} + x_{102} + x_{103} +$ $x_{116} + x_{118} + x_{125} + x_{127}$	
0.011	$x_{17} + x_{18} + x_{19} + x_{24} + x_{26} + x_{27} + x_{28} + x_{29} + x_{40} +$ $x_{46} + x_{57} + x_{64} + x_{66} + x_{79} + x_{84} + x_{102} + x_{103} +$ $x_{116} + x_{118} + x_{125} + x_{127}$	
0.01	$x_{17} + x_{18} + x_{19} + x_{21} + x_{24} + x_{27} + x_{28} + x_{29} + x_{40} +$ $x_{46} + x_{57} + x_{64} + x_{66} + x_{68} + x_{79} + x_{84} + x_{103} +$ $x_{116} + x_{118} + x_{125} + x_{127}$	
0.01	$x_{17} + x_{18} + x_{19} + x_{27} + x_{28} + x_{29} + x_{40} +$ $x_{46} + x_{57} + x_{64} + x_{66} + x_{79} + x_{84} + x_{103} +$ $x_{116} + x_{118} + x_{125} + x_{127}$	
0.01	$x_{20} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} + x_{29} + x_{40} +$ $x_{46} + x_{57} + x_{64} + x_{66} + x_{79} + x_{84} + x_{85} + x_{92} + x_{101} + x_{102} +$ $x_{116} + x_{125} + x_{127}$	
0.01	$x_{16} + x_{17} + x_{20} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} + x_{29} + x_{40} +$ $x_{46} + x_{57} + x_{64} + x_{66} + x_{79} + x_{84} + x_{85} + x_{86} + x_{101} + x_{102} +$ $x_{116} + x_{125} + x_{127}$	

Pour le DES à deux tours, il n' y a pas vraiment de surprises, les biais ont en général diminué par rapport au premier tour. On note note qu'en général la liste des fonctions qui approximent une sortie du DES est de taille assez importante.

Le DES à 3 tours

Combinaison linéaire de sortie: $x_3 + x_8 + x_{14} + x_{25} + x_{49}$ (sortie de Matsui).

ϵ	Equations obtenues	temps
0.117	$x_{16} + x_{20} + x_{35} + x_{40} + x_{46} + x_{57} + x_{68} + x_{86} + x_{101}$	2
0.195	(*) $x_{17} + x_{35} + x_{40} + x_{46} + x_{57} + x_{68} + x_{92}$	

Combinaison linéaire de sortie: $x_2 + x_{36} + x_{39} + x_{40}$.

ϵ	Equations obtenues	temps
0.095	$x_4 + x_7 + x_8 + x_{34} + x_{83} + x_{97} + x_{114} + x_{124}$	3
0.095	$x_4 + x_5 + x_7 + x_8 + x_9 + x_{34} + x_{83} + x_{97} + x_{106} + x_{114} + x_{121} + x_{124}$	
0.068	$x_4 + x_5 + x_6 + x_8 + x_9 + x_{34} + x_{66} + x_{73} + x_{83} + x_{97} + x_{106} + x_{114} + x_{121} + x_{124}$	

Combinaison linéaire de sortie: $x_{17} + x_{35} + x_{40} + x_{46} + x_{57}$.

ϵ	Equations obtenues	temps
0.008	$x_8 + x_{14} + x_{25} + x_{84} + x_{100}$	36

Combinaison linéaire de sortie : $x_3 + x_8 + x_{14} + x_{25}$.

ϵ	Equations obtenues	temps
0.0063	$x_{24} + x_{25} + x_{26} + x_{27} + x_{28} + x_{29} + x_{40} + x_{46} + x_{57} + x_{64} +$ $x_{66} + x_{68} + x_{79} + x_{84} + x_{85} + x_{92} + x_{102} + x_{116} + x_{125} + x_{127}$	
0.0068	$x_4 + x_5 + x_7 + x_8 + x_9 + x_{12} + x_{13} + x_{14} + x_{16} + x_{17} + x_{33} +$ $x_{34} + x_{35} + x_{40} + x_{46} + x_{57} + x_{65} + x_{68} + x_{73} + x_{74} + x_{83} +$ $x_{90} + x_{92} + x_{97} + x_{100} + x_{103} + x_{105} + x_{106} + x_{107} + x_{121}$	
0.0063	$x_{17} + x_{18} + x_{19} + x_{40} + x_{46} + x_{57} + x_{68} + x_{103} + x_{116} + x_{118}$	
0.0068	$x_4 + x_7 + x_8 + x_{12} + x_{15} + x_{16} + x_{33} + x_{34} + x_{35} +$ $x_{40} + x_{46} + x_{57} + x_{68} + x_{73} + x_{74} + x_{83} + x_{90} + x_{91} + x_{92} +$ $x_{97} + x_{103} + x_{107}$	221
0.0068	$x_{17} + x_{18} + x_{19} + x_{24} + x_{26} + x_{27} + x_{28} + x_{29} + x_{40} + x_{46} +$ $x_{57} + x_{64} + x_{66} + x_{68} + x_{79} + x_{84} + x_{102} + x_{103} + x_{116} +$ $x_{118} + x_{125} + x_{127}$	
0.0078	$x_{17} + x_{18} + x_{19} + x_{21} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} + x_{29} +$ $x_{40} + x_{46} + x_{57} + x_{64} + x_{66} + x_{79} + x_{84} + x_{85} + x_{102} + x_{103} +$ $x_{116} + x_{118} + x_{125} + x_{127}$	

On peut voir que l'équation de Matsui possède le meilleur biais parmi les équations que l'on a obtenu, et ce biais est exactement celui qui figure dans l'article de Matsui, on remarque que le canal dans lequel on travail n'est pas le canal binaire symétrique car on obtient plusieurs approximations affines (cf 6.3).

Le DES à 4 tours

Combinaison linéaire de sortie : $x_8 + x_{14} + x_{25}$

ϵ	Equations obtenues	temps
0.005	$x_3 + x_8 + x_{14} + x_{25} + x_{84} + x_{100} + x_{119}$	5203

Combinaison linéaire de sortie : $x_{17} + x_{35} + x_{40} + x_{46} + x_{57}$.

ϵ	Equations obtenues	temps
0.00178	$x_{16} + x_{20} + x_{40} + x_{46} + x_{57} + x_{68} + x_{86} + x_{101} + x_{113}$	
0.002	$x_{17} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{92} + x_{113} + x_{118}$	5203
0.00239	$x_{18} + x_{19} + x_{40} + x_{46} + x_{57} + x_{68} + x_{103} + x_{113} + x_{118}$	

Combinaison linéaire de sortie : $x_3 + x_8 + x_{14} + x_{25} + x_{49}$ (sortie de Matsui).

ϵ	Equations obtenues	temps
0.034	(*) $x_1 + x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} +$ $x_{81} + x_{84} + x_{98} + x_{119}$	
0.036	$x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} +$ $x_{84} + x_{119} + x_{124}$	
0.036	$x_1 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{81} +$ $x_{84} + x_{119} + x_{124}$	
0.0447	$x_1 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} +$ $x_{84} + x_{113} + x_{119} + x_{124}$	
0.0492	$x_1 + x_2 + x_3 + x_8 + x_{14} + x_{25} + x_{49} +$ $x_{84} + x_{98} + x_{113} + x_{119}$	8
0.0495	$x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} +$ $x_{81} + x_{84} + x_{98} + x_{113} + x_{119}$	
0.062	$x_1 + x_2 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} + x_{84} + x_{98} +$ $x_{113} + x_{119} + x_{124}$	
0.062	$x_1 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{81} + x_{84} + x_{119} + x_{124}$	
0.074	$x_1 + x_2 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{81} + x_{84} + x_{98} + x_{119} + x_{124}$	

Combinaison linéaire de sortie : $x_3 + x_8 + x_{14} + x_{25}$.

ϵ	Equations obtenues	temps
0.00487	$x_8 + x_{14} + x_{25} + x_{84} + x_{100} + x_{119}$	376

Pour 4 tours, on a vraiment un résultat surprenant, l'équation de Matsui n'est plus la meilleure, sur la sortie qu'il considère. De plus le biais que l'on a observé pour l'équation de Matsui est inférieur au biais qu'il a calculé par le lemme 22 [34], le biais de l'article [34] correspond à $\epsilon = 0.06$, tandis que l'observation indique $\epsilon = 0.034$. En prenant comme parametre $\epsilon = 0.003$ le programme 6.7 a retourné une centaine d'équations. On en a seulement représenté qu'une fraction.

Le DES à 5 tours

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{34} + x_{35} + x_{36} + x_{37} + x_{40} + x_{46} + x_{57}$ (sortie de Matsui).

ϵ	Equations obtenues	temps
0.00676	(*) $x_1 + x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} +$ $x_{81} + x_{84} + x_{90} + x_{98} + x_{107} + x_{119} + x_{122} + x_{124}$	
0.00660	$x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} + x_{84} +$ $x_{90} + x_{107} + x_{119} + x_{122}$	
0.00759	$x_1 + x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{81} + x_{84} + x_{90} + x_{98} + x_{107} + x_{119} + x_{122} + x_{124}$	
0.00899	$x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{81} +$ $x_{84} + x_{90} + x_{98} + x_{107} + x_{113} + x_{119} + x_{122} + x_{124}$	
0.00889	$x_1 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{81} +$ $x_{84} + x_{90} + x_{107} + x_{119} + x_{122}$	
0.0113	$x_1 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} +$ $x_{81} + x_{84} + x_{90} + x_{107} + x_{119} + x_{122}$	138
0.00943	$x_2 + x_3 + x_5 + x_8 + x_{14} + x_{25} + x_{49} + x_{81} +$ $x_{84} + x_{90} + x_{98} + x_{107} + x_{119} + x_{122} + x_{124}$	
0.0091	$x_1 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} + x_{84} + x_{90} +$ $x_{107} + x_{115} + x_{119} + x_{122}$	
0.0119	$x_1 + x_2 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{84} + x_{90} + x_{98} + x_{107} + x_{115} + x_{119} + x_{122}$	
0.0121	$x_1 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} +$ $x_{81} + x_{84} + x_{90} + x_{107} + x_{113} + x_{115} + x_{119} + x_{122}$	
0.0143	$x_1 + x_2 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} +$ $x_{81} + x_{84} + x_{90} + x_{98} + x_{107} + x_{113} + x_{115} + x_{119} + x_{122}$	
0.0094	$x_1 + x_2 + x_3 + x_8 + x_{14} + x_{25} + x_{49} + x_{84} + x_{90} + x_{98} +$ $x_{107} + x_{115} + x_{119} + x_{122} + x_{124}$	

Combinaison linéaire de sortie : $x_3 + x_8 + x_{14} + x_{25}$.

ϵ	Equations obtenues	temps
0.00142	$x_{18} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{113} + x_{118}$	3584

Combinaison linéaire de sortie : $x_8 + x_{14} + x_{25}$

ϵ	Equations obtenues	temps
0.0031	$x_{17} + x_{35} + x_{40} + x_{46} + x_{57} + x_{68} + x_{92} + x_{103} + x_{113}$	5203
0.0018	$x_{16} + x_{20} + x_{35} + x_{40} + x_{46} + x_{57} + x_{68} + x_{101} + x_{103} + x_{113}$	

Pour 5 tours on peut faire les mêmes remarques que pour 4 tours, et on constate à nouveau que le lemme 22 est très approximatif pour ce type de canal. On a trouvé de meilleures équations sur la sortie que Matsui propose, et l'observation indique pour l'équation de Matsui $\epsilon = 0.0067$ tandis que la valeur annoncée dans son article était de $\epsilon = 0.019$ [34]. Par ailleurs on a retrouvé l'équation de Kalisky-Robshaw [32] pour la combinaison linéaire de sortie $x_8 + x_{14} + x_{25}$, avec les même biais que celui qu'ils ont observé.

Le DES à 6 tours

Combinaison linéaire de sortie : $x_3 + x_8 + x_{14} + x_{25} + x_{49}$ (sortie de Matsui).

ϵ	Equations obtenues	temps
0.003042	(*) $x_8 + x_{14} + x_{25} + x_{84} + x_{87} + x_{100} + x_{119}$	381

Combinaison linéaire de sortie : $x_8 + x_{14} + x_{25}$

ϵ	Equations obtenues	temps
0.00104	$x_1 + x_2 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} + x_{84} +$ $x_{87} + x_{97} + x_{98} + x_{113} + x_{119} + x_{124}$	4132
0.00133	$x_1 + x_2 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{81} + x_{84} + x_{87} + x_{97} + x_{98} + x_{119} + x_{124}$	
0.00104	$x_1 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} +$ $x_{74} + x_{81} + x_{84} + x_{87} + x_{97} + x_{119} + x_{124}$	

Au sixième tour le biais que l'on a observé pour l'équation de Matsui est relativement proche du biais prévu par Matsui [34], quoi qu'inférieur, mais c'est le meilleur biais que l'on a observé. On a par ailleurs considéré la sortie de Kaliski-Robshaw qui a donné des équations avec des biais du même ordre.

Le DES à 7 tours

Combinaison linéaire de sortie : $x_3 + x_8 + x_{14} + x_{25} + x_{49}$ (sortie de Matsui).

ϵ	Equations obtenues	temps
0.000596	(*) $x_{16} + x_{20} + x_{40} + x_{46} + x_{57} + x_{68} + x_{71} + x_{86} + x_{101} + x_{103} + x_{113}$	
0.000760	$x_{17} + x_{40} + x_{46} + x_{57} + x_{68} + x_{71} + x_{92} + x_{103} + x_{113}$	
0.001112	$x_{18} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{71} + x_{113} + x_{118}$	
0.000864	$x_{18} + x_{19} + x_{40} + x_{46} + x_{57} + x_{68} + x_{71} + x_{113} + x_{118}$	5964
0.000895	$x_{17} + x_{19} + x_{21} + x_{40} + x_{46} + x_{57} + x_{71} + x_{92} + x_{103} + x_{113} + x_{118}$	
0.000897	$x_{17} + x_{20} + x_{40} + x_{46} + x_{57} + x_{68} + x_{71} + x_{92} + x_{101} + x_{103} + x_{113}$	
0.000799	$x_{16} + x_{17} + x_{19} + x_{40} + x_{46} + x_{57} + x_{68} + x_{71} + x_{86} + x_{92} + x_{103} + x_{113} + x_{118}$	

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{34} + x_{40} + x_{46} + x_{57} + x_{64}$.

ϵ	Equations obtenues	temps
0.001028	$x_8 + x_{14} + x_{25} + x_{65} + x_{75} + x_{84} + x_{87} + x_{100} + x_{116} + x_{119} + x_{121}$	5203

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{34} + x_{36} + x_{37} + x_{40} + x_{46} + x_{57} + x_{64}$.

ϵ	Equations obtenues	temps
0.00133	$x_8 + x_{14} + x_{25} + x_{65} + x_{75} + x_{84} + x_{87} + x_{90} + x_{100} + x_{103} + x_{116} + x_{119} + x_{121}$	5203

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{36} + x_{37} + x_{40} + x_{46} + x_{64}$.

ϵ	Equations obtenues	temps
0.001040	$x_8 + x_{14} + x_{25} + x_{65} + x_{84} + x_{87} + x_{90} + x_{100} + x_{116} + x_{119} + x_{121} + x_{123}$	5203

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{34} + x_{35} + x_{40} + x_{46} + x_{57}$.

ϵ	Equations obtenues	temps
0.000846	$x_8 + x_{14} + x_{25} + x_{75} + x_{84} + x_{87} + x_{100} + x_{119} + x_{121}$	5203

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{40} + x_{46} + x_{57} + x_{64}$.

ϵ	Equations obtenues	temps
0.000821	$x_8 + x_{14} + x_{25} + x_{65} + x_{84} + x_{87} + x_{100} + x_{116} + x_{119} + x_{121}$	5203

Combinaison linéaire de sortie : $x_{40} + x_{46} + x_{57}$.

ϵ	Equations obtenues	temps
0.000824	$x_1 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} + x_{81} + x_{84} + x_{87} + x_{97} + x_{119} + x_{124}$	5203
0.001334	$x_1 + x_2 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} + x_{81} + x_{84} + x_{87} + x_{97} + x_{98} + x_{119} + x_{124}$	
0.000815	$x_1 + x_2 + x_8 + x_{14} + x_{25} + x_{32} + x_{49} + x_{74} + x_{84} + x_{87} + x_{97} + x_{98} + x_{113} + x_{119} + x_{124}$	

Au septième tour, l'équation de Matsui n'est pas la meilleur, le biais observé est 3 fois inférieur au biais prévu par le lemme 22. On a par ailleurs pu conserver grâce à la structure du DES la sortie de Kaliski-Robshaw. Cette sortie donne de meilleurs biais que la sortie de Matsui.

Le DES à 8 tours

Combinaison linéaire de sortie : $x_1 + x_2 + x_3 + x_4 + x_5 + x_8 + x_{14} + x_{25} + x_{49}$ (sortie Matsui).

ϵ	Equations obtenues	temps
0.000171	$x_{16} + x_{20} + x_{40} + x_{46} + x_{57}$ $x_{74} + x_{76} + x_{78} + x_{81} + x_{83} + x_{84} + x_{89} + x_{90} + x_{91}$ $x_{92} + x_{94} + x_{95} + x_{97} + x_{98} + x_{99} + x_{102} + x_{106}$ $x_{107} + x_{108} + x_{117} + x_{119} + x_{121} + x_{122} + x_{127}$	29874

Combinaison linéaire de sortie : $x_{17} + x_{33} + x_{34} + x_{36} + x_{37} + x_{40} + x_{46} + x_{57} + x_{64}$.

ϵ	Equations obtenues	temps
0.000311	$x_{18} + x_{19} + x_{40} + x_{46} + x_{57} + x_{68} + x_{71} + x_{73} + x_{77} + x_{79} + x_{83} + x_{84} + x_{85} + x_{89} + x_{91} + x_{92} + x_{95} + x_{97} + x_{101} + x_{108} + x_{111} + x_{114} + x_{116} + x_{118} + x_{121} + x_{125}$	29874

Au huitième tour le temps de calcul devient important, et on a pas pu tester un grand nombre de sorties, cependant on a pu observé que l'équation de Matsui tenait avec un biais 5 fois inférieur au biais prévu par le lemme 22. On a pu cependant trouvé une sortie pour laquelle l'équation que l'on a obtenu tient avec un biais plus important.

6.6.4 Le bilan

On a obtenu de nombreuses équations par décodage. On observe qu'a partir du troisième tour jusqu'au huitième tour on a trouvé de meilleures équations que celles de Matsui, excepté pour le sixième tour. On a pu observé que le lemme 22 n'était pas très fiable dans ce type de canal. On a pu se rendre compte que le canal que l'on a rencontré était du type adversaire. Les résultats de Tor Helleseth, Torleiv Klov, Vladimir I. Levenshtein [28] permettent de dire que les canaux que l'on a rencontré sont très loins de ressembler au canal binaire symétrique. Compte tenu de ces remarques, il serait judicieux de lancer des calculs sur plus de tours de DES (9,10,11) tours car le résultat est imprédictible et risque d'être surprenant. Il faut maintenant exploiter ceci afin d'obtenir des équations sur 14, 15 ou 16 tours de Des. Observons l'équation de Matsui et les équations que l'on a trouvées sur la sortie de Kaliski-Robshaw. Ces équations peuvent être chaînées comme nous l'avons expliqué au début de ce chapitre. On obtient ainsi des approximations affines de 12 tours de DES avec un biais (ϵ) égale à 0.0000798 pour la meilleur d'après le lemme 19. On peut faire la même opération en composant 5 tours et 7 tours car les équations que l'on a obtenues le permettent, et on obtiendrait le même biais. Il reste à compléter les 2,3 ou 4 tours qui manque à ces 12 tours. Ceci est possible grâce aux approximations que l'on a obtenu sur 2,3 ou 4 tours. On obtient ainsi sur 16 tours par exemple l'équation suivante :

$$R_{16}[17] + L_{16}[1,2,4,5,8,14,25,32] + R_0[18,19,21] + L_0[8,14,25] + K[3,7,10,13,14,25,42,45,46,50,57] = 0,$$

et cette équation tient avec probabilité $1/2 + 0.91 \times 2^{-24}$, si cette probabilité est calculée grâce au lemme 22. Cette probabilité est inférieur à la probabilité de la meilleur approximation linéaire que Matsui a trouvé pour 16 tours [34], puisque son équation tient avec probabilité $1/2 - 1.49 \times 2^{-24}$. Notons que notre méthode permet d'obtenir plusieurs dizaines d'équations sur 16 tours avec des biais du même ordre.

Chapitre 7

Courbes modulaires sur $\mathbb{F}_p$

Dans ce chapitre nous nous intéresserons à la construction de courbes utiles pour former de bon codes correcteur d'erreurs, puis pour former de bons cryptosystèmes. Ce chapitre ne contient pas de résultats nouveaux. On a simplement donner une présentation originale des résultats existants, que l'on a essayé de replacer dans le contexte des applications.

Donnons nous un corps $\mathbb{F}$ de cardinal q. On sait depuis une vingtaine d'années [25], [26] que les courbes $C/\mathbb{F}$ de genre g ont au plus $(q^{\frac{1}{2}} - 1 + o(1))g$ points rationnels sur $\mathbb{F}$ lorsque $g \to \infty$ (c'est la borne de Drinfield-Vladut [25]), et que si q est un carré alors il y a une classe de courbes appelée courbes modulaires $C/\mathbb{F}$ de genre $g \to \infty$ avec $\#(C(\mathbb{F})) \geq (q^{\frac{1}{2}} - 1)(g - 1)$. Donc de telles courbes sont "assymptotiquement optimales" : elles atteignent la limite supérieur de $\#(C(\mathbb{F}))/g(C)$. Ces courbes vont donc donner d'excellents codes correcteur d'erreurs sur $\mathbb{F}$, ce sont des codes dit "géométriques". Pour construire, et donc utiliser de tels codes il faut pouvoir construire les équations explicites de ces courbes optimales [8], [26]. D'autre part par construction, la longueur de ces codes géométriques est donnée par le cardinal des jacobiennes associées à ces courbes.

On sait depuis quelques temps que les courbes elliptiques et les courbes hyperelliptiques de genre 2 peuvent être utilisée pour faire de la signature. Ces courbes formes des cryptosystèmes réputés très solides. Pour effectivement construire de tels cryptosystèmes, il faut se donner une courbe sur un corps $\mathbb{F}$, et le cardinal de la jacobienne associé sur ce corps. Donc on va décrire une méthode originale pour calculer le cardinal de la Jacobienne de certaines courbes modulaires sur un corps premier $\mathbb{F}_p$. On donnera ensuite simplement une idée de l'algorithme pour calculer un modèle affine plan de ces courbes. Ces méthodes ont été introduites entre autre par J. Cremona [2], X. Wang [51], L. Merel [36] et plus récemment par M. Müller et G. Frey [18], la suite de ce chapitre provient également de [47].

D'autre part, on sait maintenant que toutes les courbes elliptiques sont modulaires [22]. Une conjecture de Shimura-Taniyama [38] dit que toute variété abélienne est facteur directe de la jacobienne de la courbe modulaire $X_0(N)$.

Autrement dit, les courbes modulaires forment une grande famille de courbes.

7.1 Les courbes modulaires $X_0(N)$

7.1.1 Notion de surface de Riemann.

Soit X un espace topologique connexe et séparé. Un voisinage de coordonnées de X est un couple (U,z) avec U un ouvert de X et z un homéomorphisme de U sur un ouvert de $\mathbb{C}$. Deux voisinage de coordonnées (U_1,z_1) et (U_2,z_2) sont dits compatibles si la fonction

$$z_1 \circ z_2^{-1} : z_2(U_1 \cap U_2) \longrightarrow z_1(U_1 \cap U_2)$$

est holomorphe et à dérivée non nulle partout. Une famille de voisinages de coordonnées $(U_i,z_i)_{i\in I}$ est un revêtement si $X = \cup U_i$ et (U_i,z_i) est compatible avec (U_j,z_j) pour tout couple $(i,j) \in I \times I$. Deux voisinages de coordonnées sont dits équivalents si leur union est aussi un voisinage de coordonnées. Cela définit une relation d'équivalence sur l'ensemble des revêtements. La classe d'équivalence associée à cette relation d'équivalence se nomme structure complexe sur X. Un espace topologique connexe et séparé muni d'une structure complexe est une surface de Riemann.

Soit $F = (U_i,z_i)$ un revêtement de X. Une fonction $f : U \longrightarrow \mathbb{C}$ sur un ouvert U de X est holomorphe relativement à F si $f \circ z_i^{-1} : z_i(U \cap U_i) \to \mathbb{C}$ est holomorphe pour tout $i \in I$. Si f est holomorphe relativement à tout revêtement, on dit que f est holomorphe pour la structure complexe de X.

Une application $f : X \longrightarrow X'$ d'une surface de Riemann sur une autre surface de Riemann est holomorphe si $g \circ f$ est holomorphe pour toute g fonction holomorphe sur un ouvert de X'.(Voir [38])

Définition 30. *Soit $SL_2(\mathbb{Z})$, le groupe des matrices à coefficients dans $\mathbb{Z}$ et de déterminant 1. Alors $SL_2(\mathbb{Z})$ est engendré par*

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad et \quad R = \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}.$$

Pour un entier positif N on définit

$$\Gamma_0(N) = \{\alpha = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z}) \mid c \equiv 0 \ mod \ N \}$$

qui est le sous-groupe de Hecke de $SL_2(\mathbb{Z})$ de niveau N. Ce sous-groupe agit à droite sur le demi plan complexe étendu

$$\mathbb{H}^* := \mathbb{H} \cup \mathbb{Q} \cup \{\infty\},$$

de la manière suivante : on identifie H^* avec un sous ensemble de $\mathbb{P}^1(\mathbb{C})$ selon

$$\begin{array}{rcll} z & \longleftrightarrow & (z:1); & z \in \mathbb{H} \\ r & \longleftrightarrow & (r:1); & z \in \mathbb{Q} \\ \infty & \longleftrightarrow & (1:0). & \end{array}$$

L'action de $SL_2(\mathbb{Z})$ sur $\mathbb{H}^*$ est définie par son action sur $\mathbb{P}^1(\mathbb{C})$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot (z_1 : z_2) \longmapsto (az_1 + bz_2 : cz_1 + dz_2)$$

On a alors la relation d'équivalence suivante : deux éléments de $\mathbb{H}^*$ sont équivalents, si et seulement si, ces deux éléments appartiennent à la même orbite. On note alors $X_0(N)$, le quotient par cette action

$$X_0(N) := \Gamma_0(N)\backslash\mathbb{H}^*.$$

L'ensemble $\Gamma_0(N)\backslash\mathbb{Q}\cup\{\infty\}$ représente les pointes de $X_0(N)$, et $X_0(N)$ est appelé courbe modulaire de niveau N. On peut montrer que $X_0(N)$, muni de la topologie héritée de $\mathbb{C}$, qui est celle définie par

$$\{z|\Im(z) > M\}, \quad M > 0$$

qui forme un système fondamental de voisinages de l'∞, et

$$\{z|\ |z - (a + ir)| < r\} \cup \{a\},$$

et qui forme un système fondamental de voisinages de $a \in \mathbb{Q}$, forme un espace topologique compact, connexe et séparé.

7.1.2 Structure complexe sur $X_0(N)$

Soit $\pi : \mathbb{H} \to \Gamma_0(N)\backslash\mathbb{H}^*$ la surjection canonique. Pour $z_0 \in \mathbb{H}$, on choisit un voisinage V de z_0 tel que

$$\text{pour tout } \gamma \in \Gamma_0(N),\ \gamma V \cap V \neq \varnothing \Longrightarrow \gamma z_0 = z_0,$$

et soit $U = \pi(V)$, c'est un ouvert puisque $\pi^{-1}U = \cup_\gamma \gamma V$ est un ouvert.

1. Si le stabilisateur de z_0 dans $\Gamma_0(N)$ est $\pm I$, alors π est un homéomorphisme d'inverse φ et (U,φ) est un voisinage de coordonnées.
2. Si le stabilisateur de z_0 dans $\Gamma_0(N)$ est $\neq \pm I$, alors c'est un sous-groupe d'ordre 3 ou 6. Si D désigne le disque unité, soit

$$\lambda : \mathbb{H} \to D, \quad z \mapsto \frac{z - z_0}{z - \overline{z_0}},$$

alors il existe une application $\varphi : U \to \mathbb{C}$ tel que $\varphi(\pi(z)) = \lambda(z)^n$, et l'on obtient (U,ϕ) comme voisinage de coordonnées.

3. Si $z_0 = \infty$, on choisit comme voisinage de l'infini $\{z|\Im(z) > 2\}$ et soit $U = \pi(V)$ alors il existe une application bien définie $\varphi : U \to \mathbb{C}$ telle que $\varphi(\pi(z)) = e^{2\pi iz}$ et (U,φ) est un voisinage de coordonnées.
4. Si $z_0 \in \mathbb{Q}$, on choisit $\beta \in SL_2(\mathbb{Z})$ tel que $\beta(z_0) = \infty$ et on procède de façon similaire.

Pour des démonstrations plus détaillées, voir [38].

Proposition 8. *Les voisinages de coordonnées définies précédemment sont compatibles et définissent sur $X_0(N)$ une structure de surface de Riemann. Un théorème général dit qu'une surface de Riemann compacte peut être identifiée à l'ensemble des points d'une unique courbe algébrique projective sur $\mathbb{C}$. $X_0(N)$ à la remarquable propriété d'être l'ensemble des points complexes d'une courbe canonique définie sur $\mathbb{Q}$. Voir [38] .*

Exemple 43 :

1. $N = 1$, alors $X_0(1)$ a un genre nul et $X_0(1) \cong \mathbb{P}^1(\mathbb{C})$.
2. $N = 11$, alors $X_0(11)$ est une courbe de genre 1, c'est une courbe elliptique avec pour modèle de Weierstrass :

$$E : y^2 + y = x^3 - x^2 - 10x - 20.$$

(Voir [18]) ◊

7.2 Formes modulaires de poids 2 sur $X_0(N)$

Définition 31. *Une forme modulaire de poids 2 pour $\Gamma_0(N)$ est une fonction $f : \mathbb{H} \longrightarrow \mathbb{C}$ telle que :*

1. *f est holomorphe sur $\mathbb{H}$;*
2. *Pour tout $\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z})$, $f(\gamma z) = (cz + d)^2 f(z)$;*
3. *f est holomorphe sur les pointes.*

Remarque 19. *Puisque*

$$f(\gamma z) = (cz + d)^2 f(z),\ T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \in \Gamma_0(N)$$

alors

$$f(Tz) = f(z).$$

c'est-à-dire $f(z + 1) = f(z)$, et donc $f(z) = f^*(q)$ pour $q = e^{2\pi iz}$. Donc, dire que f est holomorphe en l'∞ veut dire que f^* est holomorphe en 0; on a donc

$$f(z) = \sum_{n \geq 0} c(n) q^n,\ \ q = e^{2\pi iz}.$$

Remarque 20. *Montrer que f est holomorphe en une pointe $r \neq \infty$, revient à montrer que $f\circ\gamma$ est holomorphe en l'∞ pour $\gamma \in SL_2(\mathbb{Z})$ tel que $\gamma(r) = 0$. L'ensemble des formes modulaires de poids 2 sur $X_0(N)$ est noté $M_2(N)$.*

Remarque 21. *Pour tout $\gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z})$, on a :*

$$d\gamma z = d\frac{az+b}{cz+d} = \frac{a(cz+d) - c(az+b)}{(cz+d)^{-2}}dz = (cz+d)^{-2}dz.$$

Donc si f est une forme modulaire de poids 2, $f(z)dz$ est invariante sous l'action de $\Gamma_0(N)$.

Définition 32. *Une forme modulaire qui s'annule sur les pointes s'appelle une forme parabolique. Par exemple, le fait qu'une forme parabolique s'annule en l'∞ implique que :*

$$f(z) = \sum_{n\geq 1} c(n)q^n, \quad q = e^{2\pi iz}$$

L'ensemble des formes paraboliques sur $X_0(N)$ se note $S_2(\Gamma_0(N))$ ou plus simplement $S_2(N)$.

Proposition 9. *Soit π l'application quotient $\mathbb{H}^* \longrightarrow \Gamma_0(N)\backslash\mathbb{H}^*$, et pour toute forme différentielle ω sur $\Gamma_0(N)$, on pose $\pi^*\omega = fdz$. Alors $\omega \longmapsto f$ est un isomorphisme de l'espace des 1-formes différentielles holomorphes sur $\Gamma_0(N)\backslash\mathbb{H}^*$ dans $S_2(\Gamma_0(N))$.*
Preuve : voir [38].

Corollaire 2. *Le $\mathbb{C}$-espace vectoriel $S_2(\Gamma_0(N))$ a une dimension égale au genre de $X_0(N)$.*
Preuve : C'est une version du théorème de Riemann-Roch (voir [38]).

On va maintenant donner des exemples de formes modulaires de poids k pour l'action de $SL_2(\mathbb{Z})$ qui sont appelées séries d'Eisenstein. Elles joueront un rôle important par la suite.

Définition 33. *(Séries d'Eisenstein [42],[40]). Par exemple, une série d'Eisenstein de poids k pour l'action de $SL_2(\mathbb{Z})$ est donnée par l'expression*

$$G_k(z) = 2\zeta(2k) + \frac{2(2\pi i)^{2k}}{(2k-1)!}\sum_{n=1}^{\infty}\sigma_{2k-1}(n)q^n, \text{ avec } \sigma_{2k-1}(n) = \sum_{d|n} d^{2k-1} \text{ et}$$

$$\zeta(2k) = \frac{2^{2k-1}}{(2k)!}B_k\pi^{2k}, \text{ où } B_k \text{ vérifie } zcotg(z) = 1 - \sum_{k=1}^{\infty} B_k\frac{2^{2k}z^{2k}}{(2k)!}.$$

L'ensemble des séries d'Eisenstein pour l'action de $\Gamma_0(N)$ est un $\mathbb{Z}$-module de dimension $\nu_\infty(N) - 1$ (voir [42]), on verra plus tard la signification de $\nu_\infty(N)$.

7.3 L'algèbre de Hecke $\mathbf{T}_N$

Soit $\tau \in \mathbb{H}$. On note E, la courbe elliptique $\mathbb{C}/L$ où $L = \mathbb{Z} + \mathbb{Z}\tau$. Soit C_N un sous groupe cyclique d'ordre N de $E[N]$, le groupe des points de N-torsion. Ensuite, on notera $P = (E,C_N)_\sim$ un représentant à isomorphisme près de la classe de (E,C_N). En utilisant l'interprétation modulaire des points de $X_0(N)_\mathbb{Z}$ on peut définir l'opérateur d'Atkin-Lehner qui est aussi appelé involution d'Atkin-Lehner et qui est noté W_n [5]. Soit n un entier positif qui divise N et tel que $\gcd(n,N/n) = 1$, alors l'action du n-ième opérateur d'Atkin-Lehner est donnée par

$$W_n(P) = (E/C_n,(C[n] \times C_{N/n})/C_n)_\sim.$$

Encore en utilisant l'interprétation modulaire on définit l'opérateur de Hecke [5]. Soit n ne divisant pas N, alors le n-ième opérateur de Hecke est noté T_n et son action est donnée par

$$T_n(P) = \sum_G (E/G,(C[n] \times C_{N/n})/C_n)_\sim,$$

où G représente l'ensemble des sous-groupes d'ordre n de E qui ont une intersection triviale avec $(C[n] \times C_{N/n})$. En conséquence , W_n et T_n agissent sur

1. la Jacobienne $J_0(N)$ de $X_0(N)$;
2. l'espace des formes paraboliques $S_2(N)(\mathbb{Z})$;
3. le groupe d'homologie $H_1(X_0(N),\mathbb{Z})$.

Définition 34. *L'algèbre de Hecke $\mathbb{T}_N$ de niveau N est une $\mathbb{Z}$-sous algèbre de l'anneau des endomorphisme $End_\mathbb{Z}(\Omega^1(X_0(N))_\mathbb{Z})$ qui est engendré par*

$$W_n \text{ avec } n|N,\ \gcd(n,N/n) = 1 \text{ et } T_k \text{ avec } \gcd(k,N) = 1.$$

L'algèbre de Hecke est commutative.

Théorème 39. *Les opérateurs T_n et W_n ont les propriétés suivantes :*

1. $T_{nm} = T_nT_m$ *si* $\gcd(m,n) = 1$;
2. $T_pT_{p^r} = T_{p^{r+1}} + pT_{p^{r-1}}$ *si p est un nombre premier ne divisant pas N;*
3. $T_pT_{p^r} = T_p^r$, $r \geq 1$, *si p divise N.*

Définition 35. *Une forme parabolique $f \in S_2(N)$ est une forme propre de Hecke si f satisfait*

$$T(f) = \lambda_T \cdot f \text{ pour tout } T \in \mathbb{T}_N;$$

où λ_T est la valeur propre de Hecke pour T. On note

$$E_{\lambda_T} = \{f \in S_2(N) \mid T(f) = \lambda_T \cdot f\},$$

le λ_T-espace propre où $T \in \mathbb{T}_N$ est fixé. Maintenant, on définit l'espace des vieilles formes de $S_2(N)$ de la manière suivante :

$$S_2^{old} = \left\langle g(dz) \mid g(z) \in S_2(M) \text{ avec } M|N; M \neq N; d|\frac{N}{M} \right\rangle.$$

Définition 36. *Le complémentaire orthogonal de S_2^{old} pour le produit scalaire de Peterson :*

$$\langle f,g\rangle = \int_{X_0(N)} f(z)\overline{g(z)}dxdy \quad avec\ f,g \in S_2(N),\ z = x+iy,$$

se note $S_2^{new}(N)$ et est appelé espace des nouvelles formes. Une forme parabolique f est une nouvelle forme, si et seulement si, $f(z) = q + \sum_{n\geq 2} a_n q^n$ et f est une forme propre de Hecke.

Théorème 40. *Atkin-Lehner 1970 [38]. $S_2^{new}(N)$ est stable sous l'action des opérateurs T_n, et donc $S_2^{new}(N)$ se décompose en une somme directe de sous-espaces orthogonals X_i,*

$$S_2^{new}(N) = \oplus X_i$$

où chacun des X_i est un sous-espace propre commun pour les T_n avec $\gcd(n,N) = 1$. *Chacun des X_i est stable pour l'action des T_p avec $p|N$ sur $\mathbb{C}$. Les espaces X_i sont tous de dimension 1 sur $\mathbb{C}$.*

Il est bien connu que $Hom(S_2(N),\mathbb{C})$ est un $\mathbb{T}_N \otimes \mathbb{C}$-module libre de rang 1 et $\mathbb{T}_N$ est un $\mathbb{Z}$-module libre de rang le genre de $X_0(N)$ [18].

Proposition 10. *Merel 1994 [36]. Soit R un anneau commutatif et soit $\psi \in Hom(\mathbb{T}_N,R)$, alors*

$$\sum_{n=1}^{\infty} \psi(T_n)q^n \in S_2(N)(R).$$

Nous utiliserons ces propriétés les coefficients de Fourier des formes paraboliques. Comme $\mathbb{T}_N$ est un $\mathbb{Z}$-module libre de rang fini agissant sur $S_2(N)$ on obtient :

Lemme 23. *Soit $f = q + \sum_{n=2}^{\infty} a_n q^n \in S_2^{new}(N)$ une forme propre de Hecke et $T \in \mathbb{T}_N$. Alors la valeur propre λ_T est un entier algébrique réel et le corps*

$$K_f = \mathbb{Q}(\lambda_T \mid T \in \mathbb{T}_N)$$

est une extension finie de $\mathbb{Q}$.

7.4 Théorie de Hecke sur les symboles modulaires

Considérons $H_1(X_0(N),\mathbb{Z}) = \mathrm{AB}(\Pi^1(X_0(N),z))$ qui est le groupe abélien obtenu en prenant comme générateurs tous les chemins fermés de $X_0(N)$, et en quotientant par la relation : deux chemins fermés sont équivalents s'il existe une déformation continue faisant passer de l'un à l'autre. Soit $\alpha, \beta \in \mathbb{H}^*$ des points équivalents sous l'action de $\Gamma_0(N)$, c'est-à-dire tels que $\beta = M(\alpha)$ pour un $M \in \Gamma_0(N)$, alors tout chemin lisse qui va de α vers β détermine une classe d'homologie intégrale dans $H_1(X_0(N),\mathbb{Z})$ qui dépend seulement de α et β ($\mathbb{H}^*$ est simplement connexe). On note cette classe d'homologie par le symbole modulaire

$\{\alpha,\beta\}$. Inversement toute classe d'homologie intégrale $\gamma \in H_1(X_0(N),\mathbb{Z})$ peut être représenté par un symbole modulaire $\{\alpha,\beta\}$.

Proposition 11. *Soit $\alpha,\beta,\gamma \in \mathbb{H}^*$, et soit $M \in \Gamma_0(N)$. Alors*

1. $\{\alpha,\alpha\} = 0$;
2. $\{\alpha,\beta\} + \{\beta,\gamma\} + \{\gamma,\alpha\}$;
3. $\{M\alpha,M\beta\} = \{\alpha,\beta\}$;

Corollaire 3. *L'application $M \mapsto \{\alpha,M\alpha\}$ est un morphisme de groupe surjectif $\Gamma_0(N) \longrightarrow H_1(X_0(N),\mathbb{Z})$, qui est indépendant de $\alpha \in \mathbb{H}^*$.*

On considère $H_1(X_0(N),cusp,\mathbb{Z})$, le groupe d'homologie relative de $X_0(N)$ pour l'ensemble des pointes. En particulier on peut voir que $H_1(X_0(N),\mathbb{Z})$ est un sous-groupe de $H_1(X_0(N),cusp,\mathbb{Z})$ car on peut prendre comme élément de $H_1(X_0(N),\mathbb{Z})$, une combinaison linéaire d'élément $\{\alpha,M\alpha\}$ avec $\alpha \in \mathbb{Q} \cup \{\infty\}$. On notera $\mathbb{Z}^{\nu\infty}$ l'ensemble des pointes de $\Gamma_0(N)\backslash\mathbb{Q}\cup\{\infty\}$. Un symbole modulaire $\{\alpha,\beta\}$ est un élément de $H_1(X_0(N),cusp,\mathbb{Z})$, où α, β sont des pointes. Pour $\alpha \in \mathbb{Q} \cup \{\infty\}$, on note $[\alpha]$ son image dans $\Gamma_0(N)\backslash\mathbb{Q} \cup \{\infty\}$. On étudiera par la suite plus précisément cette correspondance.

Proposition 12. *(Eichler and Shimura) [36]. On a la suite exacte*

$$\begin{array}{ccccccccccc} 0 & \to & H_1(X_0(N),\mathbb{Z}) & \to & H_1(X_0(N),cusp,\mathbb{Z}) & \overset{\delta}{\to} & \mathbb{Z}^{\nu\infty} & \overset{\theta}{\to} & \mathbb{Z} & \to & 0 \\ & & \{\alpha,M\alpha\} & \mapsto & \{\alpha,M\alpha\} & & & & & & \\ & & & & \{\alpha,\beta\} & \mapsto & [\alpha]-[\beta] & & & & \\ & & & & & & \lambda[\alpha] & \mapsto & \lambda & & \end{array} \tag{7.1}$$

Maintenant, on va donner quelques rappels sur la droite projective de $\mathbb{Z}/N\mathbb{Z}$.

Proposition 13. *La droite projective sur $\mathbb{Z}/N\mathbb{Z}$ est définie par*

$$\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}) := \{(c,d) \in (\mathbb{Z}/N\mathbb{Z})^2 | (c,d,N) = 1\}\backslash \sim$$

où

$$(c,d) \sim (c',d') \text{ si et seulement si } cd' \equiv c'd \bmod N.$$

L'application ψ

$$\Gamma_0(N)\backslash SL_2(\mathbb{Z}) \longrightarrow \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}),\ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \longmapsto (c,d) \bmod N,$$

est une bijection entre $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ et $\Gamma_0(N)\backslash SL_2(\mathbb{Z})$. Les éléments de $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ sont appelés M-symboles ou encore symboles de Manin.

Théorème 41. *(Manin 1972) [36]. Le $\mathbb{Z}$-module $H_1(X_0(N),cusp,\mathbb{Z})$ est libre et de rang égal à*

$$2genre(X_0(N)) + \nu_\infty(N) - 1.$$

Il est engendré par les symboles modulaires

$$\{\{M(0),M(\infty)\}|M \in \Gamma_0(N)\backslash SL_2(\mathbb{Z})\};$$

et on a l'isomorphisme

$$\mathbb{Z}[\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})]/(u+uS,u+uR+uR^2|u \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})) \cong H_1(X_0(N),cusp,\mathbb{Z}).$$

On notera i l'involution qui agit sur $\mathbb{H}^*$, sur les symboles de Manin et les symboles modulaires de la manière suivante:

$$i(z) = -\overline{z},\ \ i((c,d)) = (c,d)\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix},\ \ i(\{\alpha,\beta\}) = \{-\alpha,-\beta\}.$$

En restreignant la suite exacte (7.1) aux éléments invariants sous l'action de cette involution, on obtient

$$0 \longrightarrow H_1(X_0(N),\mathbb{Z})_+ \longrightarrow H_1(X_0(N),cusp,\mathbb{Z})_+ \xrightarrow{\delta_+} \mathbb{Z}_+^{\nu_\infty}$$

Pour construire une base de $H_1(X_0(N),\mathbb{Z})_+$, on doit construire la matrice de δ_+, donc une base de $\mathbb{Z}_+^{\nu_\infty}$. La méthode est similaire si l'on veut construire une base de $H_1(X_0(N),\mathbb{Z})$, on a juste à omettre l'action de l'involution.

Pour trouver une base de $H_1(X_0(N),\mathbb{Z})_+$, on utilise les relations suivantes:

1. $(c:d) + (c:d)S = (c:d) + (-d:c) = 0$;
2. $(c:d) + (c:d)R + (c:d)R^2 = (c:d) + (c+d:-c) + (d:-c-d) = 0$;
3. $(c:d) - i((c:d)) = (c:d) - (-c:d) = 0$.

Ces formules nous donnent des relations entre les élément d'un système de représentants de $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$, ensuite on a juste à indexer ce système de représentants dans une base canonique, on peut alors construire notre $\mathbb{Z}$-module quotient. La méthode est similaire pour obtenir une base de $\mathbb{Z}$-module pour $\mathbb{Z}_+^{\nu_\infty}$. On a les relations suivantes:

1. $i([\alpha]) = [\alpha]$ et $[\alpha] \equiv [\beta] \Longleftrightarrow \alpha = \pm\beta \mod \Gamma_0(N)$;
2. Pour $j = 1,2$, soient $\alpha_j = p_j/q_j$, des pointes équivalentes. Alors $s_1q_2 \equiv \pm s_2q_1 \ mod\ gcd(q_1q_2,N)$ où s_j satisfont $p_js_j \equiv 1 \mod q_j$.

Maintenant on va présenter quelques résultats sur la correspondance entre homologie et formes paraboliques.

Proposition 14. *(Merel 1994) On a les isomorphismes:*

1. $H_1(X_0(N),cusp,\mathbb{Z}) \cong \varepsilon is(\Gamma_0(N)) \oplus S_2(N) \oplus \overline{S_2(N)}$;

2. $H_1(X_0(N),\mathbb{Z}) \cong S_2(N) \oplus \overline{S_2(N)}$ *and* $H_1(X_0(N),\mathbb{Z})_+ \cong S_2(N)$;
3. $\dim H_1(X_0(N),\mathbb{Z})_+ = \dim H_1(X_0(N),\mathbb{Z})_- = g(X_0(N))$;

où $\overline{S_2(N)}$ *est l'espace des formes paraboliques anti-holomorphes,* $\mathcal{E}is(\Gamma_0(N))$ *est un espace de formes modulaires de poids 2 pour l'action de* $\Gamma_0(N)$ *qui est appelé séries d'Eisenstein. Le genre de* $X_0(N)$ *est noté* $g(X_0(N))$.

On va maintenant décrire l'action de l'algèbre de Hecke sur les symboles de Manin, et les symboles modulaires:

Proposition 15. *Pour p premier et* $p \nmid N$, *si* α, β *sont des pointes, on a*

$$T_p(\{\alpha,\beta\}) = \{p\alpha,p\beta\} + \sum_{k=0}^{p-1} \left\{ \frac{\alpha+k}{p}, \frac{\beta+k}{p} \right\}.$$

Si $p^a || N$, *alors soit* $W_p = \begin{pmatrix} p^a x & y \\ Nz & p^a t \end{pmatrix}$, *avec* $x, y, z, t \in \mathbb{Z}$, $\det(W_p) = p^a$. *Alors*

$$T_{p^a}(\{\alpha,\beta\}) = \left\{ \frac{p^a x\alpha + y}{Nz\alpha + p^a t}, \frac{p^a x\beta + y}{Nz\beta + p^a t} \right\}.$$

Pour calculer la matrice des opérateurs de Hecke agissant sur les formes paraboliques on doit être capable de convertir les symboles modulaires en symboles de Manin [2].

Si $(c : d) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$, avec le lemme de Bezout, on peut trouver $a,b \in \mathbb{Z}$ tels que $\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = 1$, donc on est capable de convertir les symboles de Manin en symboles modulaires:

$$(c : d) \longrightarrow M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \longrightarrow \{M(0),M(\infty)\} = \left\{ \frac{a}{c}, \frac{b}{d} \right\}.$$

Si l'on se donne un symbole modulaire $\left\{\frac{a}{c},\frac{b}{d}\right\}$, on a l'algorithme suivant:

$$\left\{ \frac{a}{c},\frac{b}{d} \right\} = \left\{ \frac{a}{c},0 \right\} + \left\{ 0,\frac{b}{d} \right\} \text{ et on note } \left\{ 0,\frac{b}{d} \right\} = \{0,t\}$$

Soit $[a_1,\ldots,a_n]$ la fraction continue de t, c'est-à-dire

$$a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots + \cfrac{1}{a_n}}}.$$

Si on note $C_k = [a_1,\ldots,a_k]$ alors le numérateur de p_k et le dénominateur de q_k de C_k satisfont les équations (pour $i = 3,4,\ldots,k$)

$$\begin{aligned} p_i &= a_i p_{i-1} + p_{i-2},\ p_{-1} = 0,\ p_0 = 1,\ p_1 = a_1,\ p_2 = a_1a_2 + 1, \\ q_i &= a_i q_{i-1} + q_{i-2},\ q_{-1} = 1,\ q_0 = 0,\ q_1 = 1,\ q_2 = a_2. \end{aligned}$$

où $t = \frac{p_n}{q_n}$ et on sait que $p_i q_{i-1} - p_{i-1} q_i = (-1)^i$, où $i \geq 0$. Donc on obtient que

$$\{0,t\} = \sum_{i=0}^{i=n} \{M_i(0), M_i(\infty)\} \text{ avec } M_i = \begin{pmatrix} (-1)^{i-1} p_i & p_{i-1} \\ (-1)^{i-1} q_i & q_{i-1} \end{pmatrix}.$$

On va maintenant présenter une autre méthode: l'algèbre de Hecke peut directement agir sur les symboles de Manin, d'une certaine manière on aura plus besoin des fractions continues.

Définition 37. *Soient* $M_n = \{v \in M^{2\times 2}(\mathbb{Z}) \mid \det(v) = n\}$ *et*

$$(c:d)M = \begin{cases} 0 & si\ (c:d)M \notin \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}) \\ (c:d)M & si\ (c:d)M \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}). \end{cases}$$

Pour tout entier $n \in \mathbb{Z}$, *on dira que les éléments* $\Theta_n = \sum_{M \in M_n} u_M M \in \mathbb{Z}[M_n]$ *satisfont la condition* C_n *si*

$$\sum_{M \in M_n} u_M (M(\infty) - M(0)) = (\infty) - (0).$$

Théorème 42. *(Merel 1994) Si* Θ_n *satisfait la condition* C_n *alors on a les formules suivantes pour l'action des opérateurs de Hecke et de Atkin-Lehner sur les symboles de Manin:*

$$T_n((c:d)) = \sum_{M \in M_n} u_M (c:d)M \text{ pour } \gcd(n,N) = 1,$$

$$W_n((c:d)) = \sum_{M \in M_n,\ (c:d)M \equiv (0,0) \mod n} u_M \epsilon_n(gM) \text{ pour } n|N,$$

où $g = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{Z})$ *et* $\epsilon_n(gM)$ *est l'unique élément de* $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ *congru à* $(1,0)gM \mod n$ *et à* $(0,1)gM \equiv (c:d)M \mod N/n$.

Théorème 43. *(Merel 1994) Les éléments*

$$\sum_{a>b\geq 0,\ d>c\geq 0,\ ad-bc=n} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathbb{Z}[M_n] \tag{7.2}$$

satisfont la condition C_n.

On va maintenant donner un résultat très utile qui va nous donner un algorithme pour calculer précisément une base de nouvelles formes qui sera en correspondance avec les courbes elliptiques et hyperelliptiques de genre 2.

Théorème 44. *(Merel 1994) Soit* $x \in \mathbb{Z}[\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})]$ *et* $\Theta = \sum_{M \in M_1} u_M M$ *qui satisfont la condition* C_1 *et soit*

$$\epsilon_1 : S_2(N) \longrightarrow S_2(N/n),\ \ \epsilon_1(x) = \sum u_M x M, \tag{7.3}$$

pour n divisant N et où la somme est restreinte aux matrices M telles que $xM \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$. Alors x appartient à $S_2^{new}(N)$, si et seulement si, x et $W_N(x)$ appartiennent au noyau de ϵ_1 pour tout diviseur n de N.

En utilisant (7.2) et (7.3), on peut voir que la somme Θ est restreinte à la matrice identité. Avec ces résultats on est maintenant capable de construire une base de nouvelles formes qui est en correspondance avec une variété abélienne de genre 1 ou 2. Nous allons étudier cette correspondance dans la section suivante.

7.5 Nouvelles formes et variétés abéliennes

Théorème 45. *Soit $f = q + \sum_{n=2}^{\infty} a_n q^n$ une forme propre de Hecke et soit $K_f = \mathbb{Q}(a_n \mid n \in \mathbb{N})$ le corps engendré par les coefficients de Fourier de f. Alors il existe une sous-variété abélienne A_f de $J_0(N)$ et un isomorphisme $\theta : K_f \to End(J_0(N)) \otimes \mathbb{Q}$ ayant les propriétés suivantes :*

1. $\dim(A_f) = [K_f : \mathbb{Q}] = d$;
2. *Si* $\gcd(n,N) = 1$, *alors $\theta(a_n)$ coïncide avec la restriction de T_n à A_f;*
3. *Le conducteur $N(A_f)$ est égal à N^d, où $d = \dim(A_f)$.*

De plus le couple (A_f,θ) est unique et A_f est une variété abélienne simple définie sur $\mathbb{Q}$.

7.5.1 L-series et applications

Cas des courbes elliptiques

Rappellons que pour une courbe elliptique E définie sur $\mathbb{Q}$, on peut écrire

$$L(E,s) = \prod_{p \text{ bon}} \frac{1}{1 - a_p p^{-s} + p^{1-s}} \cdot \prod_{p \text{ mauvais}} \frac{1}{1 - a_p p^{-s}} = \sum a_n n^{-s}$$

où

$$a(p) = \begin{cases} 1 + p - N_p & \text{si } E \text{ a bonne réduction en } p \\ 1 & \text{si } E \text{ a une réduction multiplicative rationnelle} \\ -1 & \text{si } E \text{ a une réduction multiplicative non rationnelle} \\ 0 & \text{si } E \text{ a une réduction additive} \end{cases}$$

Rappelons qu'a toute nouvelle forme f on peut associer une série de Dirichlet qui admet un produit d'Euler [38]

$$L(f,s) = \prod_{\gcd(p,N)=1} \frac{1}{1 - a_p p^{-s} + p^{1-s}} \cdot \prod_{p|N} \frac{1}{1 - a_p p^{-s}} = \sum a_n n^{-s}$$

Théorème 46. *(Eichler-Shimura) Soit $f = q + \sum_{n=2}^{\infty} a_n q^n$ une nouvelle forme avec $a_n \in \mathbb{Z}$ pour tout $n \geq 0$. Alors, il existe une courbe elliptique E_f de conducteur N telle que $L(f,s) = L(E,s)$.*

En fait on sait que toutes les courbes elliptiques sont modulaires, c'est-à-dire que toutes les courbes elliptiques de conducteur N sont facteurs simples de la Jacobienne $J_0(N)$.

Cas des variétés abéliennes de genre 2

Soit $f = q + \sum_{n=2}^{\infty} a_n q^n$ une forme propre de Hecke, avec $K_f(a_n \mid n \in \mathbb{Z})$ une extension quadratique de $\mathbb{Q}$. Soit $I_f = \{Id,\sigma\}$ l'ensemble des plongements distincts de K_f dans $\mathbb{C}$, alors on définit la L-série de f en p par

$$L_p(f,s) = \begin{cases} 1 - a_p s + p s^2 & \text{si } p \text{ ne divise pas } N, \\ 1 - a_p s & \text{si } p \text{ divise } N. \end{cases}$$

Théorème 47. *Soit $L_p(A_f,s)$ la L-serie de A_f en p. Alors pour p premier ne divisant pas N, on a les propriétés suivantes :*

1. $L_p(A_f,s) = \prod_{\sigma \in I_f} L_p(f^\sigma,s)$;
2. $L_p(A_f,1) = \#(A_f \otimes \mathbb{F}_p)$.

En particulier on a la formule suivante :

$$L_p(A_f,1) = (1 + p + a_p)(1 + p + \sigma(a_p)) = \chi_p^f(p+1),$$

où χ_p^f est le polynôme minimal de T_p agissant sur f.

Remarque 22. *Les mêmes propriétés tiennent si $K_f(a_n \mid n \in \mathbb{Z})$ est une extension de $\mathbb{Q}$ de degré plus grand mais nous nous sommes seulement intéressés aux courbes elliptiques et hyperelliptiques de genre 2.*

7.6 Calcul du cardinal de la Jacobienne sur $\mathbb{F}_p$

On va résumer par les points suivants comment calculer le nombre de points d'une courbe elliptique ou le cardinal de la Jacobienne d'une courbe hyperelliptique de genre 2 sur $\mathbb{F}_p$:

- premièrement on construit un système de représentant de $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$, on peut prendre tous les éléments (d,i), avec d qui divise N et $\gcd(d,i) = 1$, ensuite on doit choisir un représentant dans la classe de (d,i) où d est fixé car deux éléments (d,i) et (d,j) sont équivalents si et seulement si $i - j \equiv 0 \mod N/d$;
- deuxièmement on doit déterminer une base de symbole de Manin de $H_1(X_0(N),cusp,\mathbb{Z})_+$, pour faire cela les relations que l'on a vues précédemment sont essentielles :
  1. $(c : d) + (-d : c) = 0$;

2. $(c:d)+(c+d:-c)+(d:-c-d)=0$;
3. $(c:d)-(-c:d)=0$.

On indexe juste le système de représentants de $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ par les éléments d'une base canonique, ensuite on reconnaît les relations que l'on a vues précédemment dans cette base canonique. On obtient en faite un morphisme $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}) \to H_1(X_0(N),cusp,\mathbb{Z})_+$. Pour faire ceci :

1. On extrait d'abord un système de représentants des pointes qui provient de $H_1(X_0(N),cusp,\mathbb{Z})_+$, on a vu que c'était possible car on peut convertir un symbole de Manin en symbole modulaire.
2. Ensuite on utilise les propriétés d'équivalence entre pointes [2] :
   (a) $i([\alpha])=[\alpha]$ et $[\alpha]\equiv[\beta] \Longleftrightarrow \alpha=\pm\beta \bmod \Gamma_0(N)$;
   (b) Pour $j=1,2$, soient $\alpha_j=p_j/q_j$, des pointes équivalentes . Alors $s_1q_2\equiv\pm s_2q_1 \bmod gcd(q_1q_2,N)$ où les s_j satisfont $p_js_j\equiv 1 \bmod q_j$.

On obtient aussi un morphisme cusps $\longrightarrow \mathbb{Z}_+^{\nu_\infty}$

- Maintenant on est capable de construire la matrice de δ_+ car on a une base de $\mathbb{Z}_+^{\nu_\infty}$ et $H_1(X_0(N),cusp,\mathbb{Z})_+$ avec l'algorithme d'Euclide étendu, on convertit juste les symboles de Manin en symboles modulaires et on extrait les deux pointes de chaque symbole modulaire.
- Pour obtenir une base de symbole de Manin de $S_2(N)$, on calcule juste le noyau de δ_+. Donc on obtient les vecteurs de base. On obtient la base de symboles de Manin en regardant l'indexation que l'on a choisie pour le système de représentants de $\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$.
- Notre but maintenant est de nous restreindre à une base de nouvelles formes qui est en correspondance avec les variétés abéliennes de genre 1 ou 2. Donc on va choisir le plus petit nombre premier p ne divisant pas N, et on calcule la matrice du p-ième opérateur de Hecke T_p agissant sur $S_2(N)$. Ensuite, on calcule le polynôme caractéristique de T_p et on extrait une base de vecteurs propres qui correspondent aux facteurs irréductibles de degré 1 et 2 du polynôme caractéristique de T_p.
- Pour chaque vecteur propre, on vérifie que c'est un élément de $S_2^{new}(N)$ avec l'application (7.3) $\epsilon_1 : S_2(N) \longrightarrow S_2(N/n)$ avec $n \mid N$. On garde seulement les éléments qui appartiennent à $S_2^{new}(N)$. Donc, on obtient une base de symbole de Manin de $S_2^{new}(N)$ qui sont en correspondance avec les variétés abéliennes de genre 1 ou 2. Bien sûr parfois il n'existe pas de variétés abéliennes de genre 1 ou 2 pour le niveau N qui nous est donné. On s'intéresse juste au meilleur cas où il y a au moins une variété abélienne de genre 1 ou 2.
- Maintenant on aimerait calculer les coefficients de Fourier de ces nouvelles formes afin d'obtenir le cardinal de la Jacobienne de ces variétés sur $\mathbb{F}_p$. En fait les valeurs propres du p-ième opérateur de Hecke qui agit sur les éléments de base de $S_2^{new}(N)$ donnent précisément le p-ième coefficient de

la nouvelle forme qui est en bijection avec l'élément de base sur lequel l'opérateur de Hecke a agi. Donc, pour calculer la série d'une nouvelle forme il faut juste calculer les valeurs propres de l'algèbre de Hecke agissant sur les éléments de base de $S_2^{new}(N)$.

Dans cet exemple, on a programmé cet algorithme avec le logiciel Magma[1] [47]:
voici un système de représentants de $\mathbb{P}^1(\mathbb{Z}/33\mathbb{Z})$. On choisit l'ordre naturel donné par Magma pour former une base canonique:

```
>RepresSyst(33);

[
    [ 1, 0 ],[ 1, 1 ],[ 1, 2 ],[ 1, 3 ],[ 1, 4 ],[ 1, 5 ],..
    ,[ 1, 32 ],[ 3, 14 ],[ 3, 11 ],[ 3, 20 ],..,[ 3, 1 ],
    [ 11, 3 ],[ 11, 2 ],[ 11, 1 ],[ 0, 1 ]
]
```

Maintenant, voici une base de symboles de Manin de $S_2(33)$. On sait que le genre de $X_0(N)$ est égal à 3., donc on obtient 3 vecteurs. On prend également l'ordre naturel pour indexer la base de $S_2(33)$.

```
>S2base(33);

  [                [                [
    -[ 3, 5 ]        [ 3, 5 ]         -[ 3, 4 ]
    -[ 11, 3 ]      +[ 3, 4 ]         -[ 11, 3 ]
    +[ 11, 1 ]      -[ 3, 1 ]         +[ 11, 1 ]
  ],                +[ 11, 3 ]     ]
                    -[ 11, 1 ]
                ],
```

Le plus petit nombre premier ne divisant pas N est 2. Donc on calcule l'action du 2-ième opérateur de Hecke sur $S_2(33)$, car on veut extraire les nouvelles formes qui nous intéressent.

```
> HeckeAction(2,33);

      [ 0  2  1]
A :=  [ 0 -2  0]
      [ 2  2 -1]
```

```
> CharcPolyHecke(2,33);

[
  <x - 1, 1>,
  <x + 2, 2>
]
```

1. http://www.magma.maths.usyd.edu.au/magma/

On voit qu'il y a deux espaces propres, on peut les associer aux valeurs propres respectives 1 et -2. On a maintenant besoin des vecteurs propres associés à ces valeurs propres :

```
> Eigenspace(A,-2);
Echelonized basis:
( 1  0 -1)
( 0  1  0)
```

```
> Eigenspace(A,1);
Echelonized basis:
(2 2 1)
```

On cherche les éléments de $S_2^{new}(N)$. Le degré des facteurs irréductibles du polynôme caractéristique de T_2 est 1. Donc si $S_2^{new}(N) \neq 0$, les nouvelles formes sont en correspondance avec les courbes elliptiques (à isogénie près) de conducteur égal à 33. On vérifie que la valeur propre 1 satisfait la condition du théorème 43 (7.3), c'est-à-dire que Eigenspace(A,1) doit appartenir au noyau de l'application ϵ_1 pour les diviseurs 3 et 11 de 33.

```
> Epsilon1(A,1,3);
(0)
```

```
> Epsilon1(A,1,11);
(0)
```

```
> Epsilon1(W33(A,1),3);
(0)
```

```
> Epsilon1(W33(A,1),11);
(0)
```

Epsilon1(A,i,3), pour $i = 1, -2$ est toujours égale à 0 car $\dim(S_2(3)) = 0$ tandis que $\dim(S_2(11)) = 1$. On vérifie que les autres vecteurs propres qui sont associés à la valeur propre -2 n'appartiennent pas à $S_2^{new}(N)$:

```
> Epsilon1(A,-2,3);
(0)
(0)
```

```
> Epsilon1(A,-2,11);
 (-4)
 (1)
```

En conséquence, le vecteur propre qui est associé à la valeur propre 1 de l'opérateur de Hecke T_2 appartient à $S_2^{new}(N)$. Donc, le 1-vecteur propre est en fait une nouvelle forme pour laquelle on peut calculer ses coefficients de Fourier. On voit que les éléments qui sont en correspondance avec la valeur propre -2 appartiennent à $S_2^{old}(N)$ car $\dim(E_{-2}) = 2$.

On a deux possibilités pour calculer les coefficients de Fourier, on peut appliquer les p-opérateurs de Hecke directement sur le 1-vecteur propre formé de symboles de Manin en utilisant les résultats de Manin et Merel, ou on peut transformer ces symboles de Manin en symboles modulaires et on utilise les fractions continues. En pratique, les fractions continues sont plus faciles à implémenter.

$N = 33$: genre$(X_0(33)) = 3$, on peut obtenir une nouvelle forme associée à une courbe elliptique :[43]

$$f(z) = q + q^2 - q^3 - q^4 - 2q^5 - q^6 + 4q^7 - 3q^8 + q^9 - 2q^{10} + q^{11} + q^{12} - 2q^{13} \ldots$$

Cette courbe admet pour modèle minimal $E : y^2 + xy = x^3 + x^2 - 11x$ [2]

7.7 Construction des courbes modulaires

On va d'abord résumer les résultats de Shimura [42]. Soit $f(z)$ une nouvelle forme de poids 2 et $\omega(f) = 2\pi i f(z)dz$ la différentielle associée. Soient $I_f = \{\sigma_1, \ldots, \sigma_d\}$ tous les plongements distincts de $K_f = \mathbb{Q}(a_1, \ldots)$ dans $\mathbb{C}$ qui est le corps engendré par les coefficients de f. Soit $\{f^{\sigma_1}, \ldots f^{\sigma_d}\}$ l'ensemble des nouvelles formes conjuguées de f sur $\mathbb{Q}$. Il existe une variété abélienne A_f rationnelle sur $\mathbb{Q}$ (voir théorème 45) tel que l'espace des 1-formes différentielles $\Omega^1(A_f)$ est isomorphe à $\sum_{\sigma \in I_f} \mathbb{C}\omega(f^\sigma)$. Soit $\mathbf{f} = (f^{\sigma_1}, \ldots, f^{\sigma_d})^t$ et $\omega(\mathbf{f}) = (\omega(f^{\sigma_1}), \ldots, \omega(f^{\sigma_d}))^t$. Alors l'image de $H_1(X_0(N),\mathbb{Z})$ par l'application

$$H_1(X_0(N),\mathbb{Z}) \longrightarrow \mathbb{C}^d;\ \gamma \longmapsto \int_\gamma \omega(\mathbf{f}) = \left(\int_\gamma \omega(f^{\sigma_1}), \ldots, \int_\gamma \omega(f^{\sigma_d}) \right)^t$$

est un $\mathbb{Z}$-module libre de rang $2d$. C'est donc un réseau Λ_f de $\mathbb{C}^d$ et on a

$$A_f \cong \mathbb{C}^d/\Lambda_f.$$

Quand $d = 1$ on obtient une courbe elliptique et dans ce cas il est possible d'obtenir un modèle minimal tel que $C \cong A_f$ (voir [2]).

Quand $d = 2$, parfois on peut obtenir un modèle de courbe hyperelliptique de genre 2 C, telle que $Jac(C) \cong A_f$. Ce modèle peut être obtenu si la période de la matrice satisfait certaines conditions (voir [51]).

7.8 Conclusion

Avec ces méthodes, il est possible de construire une famille très générale de courbes parce que l'on sait par exemple que toute courbe elliptique est modulaire. En fait pour un niveau N donné on est capable de construire à isogénie près, toutes les courbes elliptiques de conducteur N définies sur $\mathbb{Q}$. La complexité de cet algorithme est polynomiale en N. Donc, si le paramètre N n'est pas trop grand, on peut obtenir un grand nombre de variétés abéliennes. La conjecture de Shimura-Taniyama dit que toute variété abélienne définie sur $\mathbb{Q}$, à multiplication réelle est isogène à un facteur de $J_0(N)$, pour un N convenable. Malheureusement, avec cet algorithme il n'est pas possible de calculer le cardinal des Jacobiennes de courbes elliptiques et hyperelliptiques de genre 2 sur $\mathbb{F}_p$ lorsque p atteint une taille cryptographique ($p \approx 2^{180}$ pour une courbe elliptique et $p \approx 2^{90}$ pour le genre 2). Si l'on choisit la méthode des fractions continues, alors il faut déterminer p fractions continues. Ce coût est de l'ordre de $\mathcal{O}(p\log(p))$ opérations arithmétiques. La méthode qui utilise des sommes formelles de matrice qui agissent directement sur les symboles de Manin n'est pas meilleure car on sait que ces familles de matrices sont de cardinal de l'ordre de $\mathcal{O}(p\log(p))$ (voir [36], [18]) et ce n'est pas

facile en pratique de construire ces sommes. Donc on ne peut pas utiliser ces méthodes dans le cadre de la signature électronique. Pour construire des codes ces méthodes suffisent on n'a pas besoin en pratique de code de longueur 2^{180}. Une amélioration serait possible si l'on était capable de calculer la matrice du p-ième opérateur de Hecke pour des valeurs de p très grandes, ce qui serait possible si l'on pouvait par exemple calculer l'action des opérateurs de Hecke modulo des petits nombres premiers l_i avec une complexité polynômiale en l_i (CRT).

Notations

n	:	longueur du code considéré.
p	:	nombre premier.
q	:	nombre de la forme 2^t.
$\mathbb{F}_p$	:	corps premier de caractéristique p première.
$\mathbb{F}_q$	:	corps de caractéristique 2 du type $\mathbb{F}_{2^t}$.
d	:	distance de Hamming.
Δ	:	distance de Hamming normalisée.
M	:	distance définie par $\max(\Delta, 1-\Delta)$.
$\mathbb{F}_p[x]$	:	anneaux des polynômes sur le corps $\mathbb{F}_p$.
$\mathbb{F}_q[x]$	:	anneaux des polynômes sur le corps $\mathbb{F}_q$.
$\mathbb{F}_p^k[x]$	:	ensemble des polynômes sur le corps $\mathbb{F}_p$ de degré au plus k.
$\mathbb{F}_q^k[x]$	:	ensemble des polynômes sur le corps $\mathbb{F}_q$ de degré au plus k.
$\mathbb{F}_p[x_1,\ldots,x_m]$	:	anneaux des polynômes m-varié sur le corps $\mathbb{F}_p$.
$\mathbb{F}_q[x_1,\ldots,x_m]$	:	anneaux des polynômes m-varié sur le corps $\mathbb{F}_q$.
$\mathbb{F}_p^r[x_1,\ldots,x_m]$	:	ensemble des polynômes m-varié sur le corps $\mathbb{F}_p$ de degré total au plus r.
$\mathbb{F}_q[x_1,\ldots,x_m]$	:	anneaux des polynômes m-varié sur le corps $\mathbb{F}_q$.

$\mathbb{F}_p^r[x_1,\ldots,x_m]$	:	ensemble des polynômes m-varié sur le corps $\mathbb{F}_p$ de degré total au plus r.
$\mathbb{F}_q^r[x_1,\ldots,x_m]$	:	ensemble des polynômes m-varié sur le corps $\mathbb{F}_q$ de degré total au plus r.
f	:	fonction du type $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$ pour $m \geq 1$.
Prob	:	probabilité.
ϵ	:	parametre qui exprime la distance normalisée $\frac{1}{2} - \epsilon$. entre 2 fonctions ou 2 mots d'un code.
m	:	nombre de variables d'une fonction du type $\mathbb{F}_q^m \rightarrow \mathbb{F}_q$.
$\varkappa(x)$	:	densité normale de probabilité.
$\mathcal{N}(x)$	:	distribution normale de probabilité.
$RS_q[q-1,k]$	:	code de Reed-Solomon de longueur $q-1$ et de dimension k sur $\mathbb{F}_q$.
$RS_q[n,k]$	:	code de Reed-Solomon de longueur n et de dimension k sur $\mathbb{F}_q$.
$RM[r,m]$	:	code de Reed-Muller de longueur $2^m - 1$ et d'ordre r sur l'alphabet $\mathbb{F}_2$.
$RM_q[r,m]$	:	code de Reed-Muller de longueur $q^m - 1$ et d'ordre r sur l'alphabet $\mathbb{F}_q$.
C	:	code.
$\mathcal{G}$	:	matrice génératrice d'un code.
$\mathcal{H}$	:	matrice de parité d'un code.

Bibliographie

[1] A. Canteaut. Cryptanalyse différentielle et cryptanalyse linéaire des chiffrements de type DES, 1999.

[2] John Cremona. *Algorithms of modular elliptique curves.* Cambridge University Press, Cambridge, 1997.

[3] I. Dumer. Recursive decoding of Reed-Muller codes. In *Proc. 37th Allerton Conf on Commun.,Cont., and Comp., Monticello,IL*, pages 61–69, Sept. 22-24 1999.

[4] I. Dumer. Decoding of low-rate Reed-Muller codes with polylog. complexity. In *WCC*, 2003. url : http://www-rocq.inria.fr/codes/WCC2003/dumer.ps.

[5] Bas Edixhoven. The modular curves $X_0(N)$. In *Trieste, ICTP, Summer school on elliptic curves*, 1997.

[6] L. R. Welch et E. R. Berlekamp. Error correction of algebraic block codes. *US Patent*, 4(633):470, Decembre 1986.

[7] T. Johansson et F. Jönsson. Fast correlation attacks thought reconstruction of linear polynomials. In Springer-Verlag, editor, *CRYPTO*, number 1880 in LNCS, pages 300–315, Berlin Heilderberg, 2000.

[8] Michael A. Tsfasman Serge G. Vladuts et Gregory L. Katsman. Modular curves and codes with polynomial construction. *IEEE Trans Inform. Theory*, 2:353–355, 1984.

[9] Rudolph Lidl et Harald Niederreiter. *Finite Fields*, volume 20. Cambridge University Press, 1997.

[10] J. Von Zur Gathen et J. Gerhard. *Modern compute algebra.* Cambridge University Press, 1999.

[11] C. Harpes G. Kramer et J. Massey. generalization of linear cryptanalysis and the applicability of matsui's piling-up lemma. In *Eurocrypt*, Lectures notes in computor science. Springer, 1995.

[12] J.W. Cooley et J.W. Turkey. An algorithm for the machine computation of the complex fourier series. *Mathematics of computation*, 19:297–301, Avril 1965.

[13] O. Goldreich et L. A. Levin. A hard-core predicate for all one-way function. In *21 st Annual ACM Symposium on Theory of Computing*, pages 25–32, Newyork, 1989. ACM Press.

[14] Kaisa Nyberg et Lars R. Knudsen. Provable security against differential cryptanalysis. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 566–574, Santa Barbara, California, USA, Aout 1993. Springer.

[15] Jonathan Katz et Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes (2000). In *32th Annu. ACM Symp. on the Theory of Computing*, pages 80–86, 2000. url = "citeseer.nj.nec.com/katz00efficiency.html".

[16] O. Goldreich R. Rubinfeld et M. Sudan. Learning polynomials with queries: the highly noisy case. In *36th Annual Symposium on Fondation of Computor Science*, pages 294–303, Milwaukee, Wisconsin, 23-25 Octobre 1995.

[17] Venkatesan Guruswami et Madhu Sudan. Improved decoding of reed-solomon codes and algebraic geometry codes. In *IEEE Transactions on Information Theory*, volume 6, pages 1757–1767, 8-11 November 1999.

[18] Gerhard Frey et Michael Müller. Arithmetic of modular curves and application. *Algorithmic Algebra and Number Theory, Springer*, 1998.

[19] D. Amgluin et P. Laird. Learning from noisy examples. In *Machine Learning*, volume 2, pages 343–370, 1988.

[20] T. Høholdt et R. Refslund Nielsen. Decoding Reed-Solomon codes beyond half the minimum distance. In *International Conference on Coding Theory, Cryptography and Related Areas*, 1998.

[21] D. Ron et R. Rubinfield. Learning fallible deterministic finite automata. In *Machine Learning*, volume 18, pages 149–185, 1995.

[22] C. Breuil B. Conrad F. Diamond et R. Taylor. On the modularity of elliptic curves over q. *J. Am. Math Soc*, 14(4):843–939, 2001.

[23] Vera S. Pless W. Cary Huffman et Richard A. Brualdi. *dans "Handbook of Coding Theory", An Intoduction to Algebraic Codes.* Elsevier Science, 1998.

[24] M. A. Tsfasman et S. G. Vladut. Algebraic-geometric codes. *Dordrecht: Kluwer*, 1991.

[25] S. G. Drinfeld et S. G. Vladut. The number of points of an algebraic curve. *Functional Anal. Appl*, 17(1):53–54, 1983.

[26] Michael A. Tsfasman Serge G. Vladuts et Th. Zink. Modular curves, shimura curves, and goppa codes, better than varshamov gilbert bound. *Math. Nachr.*, 109:21–28, 1982.

[27] J. Daemen et V. Rijmen. AES proposal : Rijndael, 1999.

[28] Tor Helleseth Torleiv Klov et Vladimir I. Levenshtein. Bounds based on monotone structure of correctable and uncorrectable errors. In G.Kabatianski D.Augot, P.Charpin, editor, *WCC 2003*, page 243, Versailles (France), Mars 2003. INRIA et ENSTA.

[29] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. Wiley, troisième edition, 1950.

[30] T. Jakobsen. Cryptanalysis of block ciphers with probabilistic non linear relations of low degree. In H. Krawczyk, editor, *Crypto'98*, number 1462 in LNCS, pages 347–362. Springer, 1998.

[31] Pascal Junod. On the complexity of matsui's attack. In *ASIACRYPT, Rump Session*, pages 3–7, Kyoto, Japan, December 2000.

[32] Kaliski-Robshaw. Linear cryptanalysis using multiple approximations. In *Advances in Cryptology - CRYPTO'94*, number 839 in Lecture Notes in Computer Science, pages 27–40. Springer-Verlag, 1995.

[33] M. Kiwi. Testing and weight distribution of dual codes. In *Technical Report TR-97-010*, http://www.eccc.uni-trier.de/eccc/, 1997. Electronique Colloquium on Computation Complexity (E-CCC).

[34] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - EUROCRYPT'93*, number 765 in Lecture Notes in Computer Science, pages 386–397. Springer-Verlag, 1994.

[35] M. Matsui. The first experimental cryptanalysis of the data encryption standard. In *Advances in Cryptology - CRYPTO'94*, number 839 in Lecture Notes in Computer Science, pages 1–11. Springer-Verlag, 1995.

[36] Loïc Merel. Universal fourier expansions of modular forms. *Lecture Notes in Mathematics*, 1994.

[37] Jean-Francois Mestre. Construction de courbes de genre 2 à partir de leur modules. *Effective Methods in Algebraic Geometry*, 1991.

[38] Joseph Milne. Elliptic curves, 1996.

[39] Rasmus Refslund Nielsen. *Decoding AG codes beyond half the minimum distance.* PhD thesis, Technical University of Denmark, 1998. Sur le Web http://www.student.dtu.dk/~p938546/public.html.

[40] Jean-Pierre Serre. *Cours d'arithmétique.* Presses Univ. France, 1970.

[41] C. E. Shannon. The communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949.

[42] G. Shimura. Introduction to the arithmetic theory of automorphic functions. *Lecture Notes in Mathematics*, 1971.

[43] William A. Stein. The modular forms database. Technical report, http://www.modular.fas.harvard.edu/Tables/index.html, 1999.

[44] M. Sudan. *Efficient Checking of Polynomial and proofs and the hardness of Approximation Problems.* PhD thesis, University of California, Berkeley, 1992.

[45] M. Sudan. Decoding reed solomon codes beyond the error-correction bound. *journal of complexity*, 13:180–193, Mars 1997.

[46] M. Sudan. Improved low degree testing and its application. In *Technical Report*, http://www.eccc.uni-trier.de/eccc/, 1997. Electronique Colloquium on Computation Complexity (E-CCC).

[47] Cédric Tavernier. Construction of modular curves and determination of their cardinality over $\mathbb{F}_p$. *journal : Finite Fields and their Applications*, Septembre 2001.

[48] Cédric Tavernier. Low degree tests or distance to Reed-Solomon codes. In *ACCT8*, Russie, Septembre 2002.

[49] A. J. Menezes P. C. van Oorschot et S. A. Vanstone. *Handbook of Applied cryptography.* CRC Press, 1997.

[50] M. Videau. Généralisation de la cryptanalyse différentielle et critère de sécurité des chiffrements à clé secrète. Technical report, INRIA, 2001.

[51] Xiangdong Wang. 2-dimensional simple factors of $J_0(N)$. *Manuscripta Mathematica*, 1995.

[52] Xiangdong Wang. The Hecke operators on the cusp forms of $\Gamma_0(N)$. *Lecture notes in Mathematics*, 1995.

[53] F. J. Mac williams et N. J. A. Sloane. *The Theory of Error Correcting Codes.* North-Holland, 1977.

ÉCOLE POLYTECHNIQUE

EDX

RAPPORT DE SOUTENANCE
DOCTORAT DE L'ECOLE POLYTECHNIQUE

Nom : TAVERNIER
Prénom : Cédric
Domaine :
DEA de rattachement :

Sujet : Tatonnements, problèmes de reconstruction univariés, multivariés, et application à la cryptanalyse des DES.
Date de soutenance :

Président du jury :	- Grigory Kabatiansky	Signature :
Membres du jury :	- CHARPIN Pascale	Signature :
	- GILBERT Henri	Signature :
	- AUGOT Daniel	Signature :
	- ZÉMOR Gilles	Signature :
	- ENGE Andreas	Signature : Andreas Enge
	-	Signature :
	-	Signature :

Rapport (2) : Le sujet principal de cette thèse est un problème central de la théorie des codes correcteurs, à savoir la reconstruct de polynômes multivariés à partir de données bruitées. Il s'agit d'un problème par ailleurs très pertinent en cryptologie et en informatique théorique. Par sa présentation claire et bien struct le candidat a su ~~expliquer~~ expliquer des concepts difficiles, faisant la preu de ses compétences pédagogiques. Le candidat a su répondre de

DÉCISION DU JURY, MENTION ACCORDÉE :

~~Honorable~~ **Très honorable** ~~Très honorable avec félicitations~~

(2) Si nécessaire, utiliser le verso

manière convaincante aux questions, montrant sa maîtrise du sujet.

En conséquence, le jury, à l'unanimité, décide d'accorder à M. Cédric Tavernier le grade de docteur de l'Ecole Polytechnique, avec la mention très honorable.

Résumé

Plusieurs thèmes sont abordés dans ma thèse, dont l'étude théorique et pratique des testeurs de bas degrés. Cette problématique se situe dans le contexte des polynômes univariés et multivariés. Les théoriciens de la complexité se sont posé la question suivante : si $f : \mathbb{F}^m \longrightarrow \mathbb{F}$ désigne une fonction (sur un corps fini $\mathbb{F}$), alors n'est-il pas plus facile de tester s'il existe au moins un polynôme de degré au plus k qui coïncide avec f sur une fraction d'au moins δ de ses entrées sans pour autant reconstruire ce ou ces polynômes, plutôt que de construire directement d'éventuels polynômes satisfaisant cette même condition ? Bien que cette question soit intéressante, les testeurs qui ont été développés sont largement inefficaces pour des paramètres cryptographiques.

La deuxième partie de cette thèse concerne alors naturellement l'étude théorique et le développement d'algorithmes de décodage dans le contexte particulier de la cryptographie. Nous avons repris les travaux de O. Goldreich, R. Rubinfeld, M. Sudan concernant le problème de reconstruction multivariée en grandes longueurs. En particulier nous nous sommes intéressés au problème du décodage des codes de Reed-Muller d'ordre 1 en grandes longueurs. O. Goldreich, R. Rubinfeld, M. Sudan donnent un algorithme de décodage pour le Reed-Muller d'ordre 1 qui est applicable non seulement dans le canal binaire symétrique, mais aussi dans le canal adversaire. Nous proposons un nouvel algorithme dont la complexité est proche des bornes prévues par le théorie de l'information dans le canal binaire symétrique. Nous étendons ensuite cet algorithme au Reed-Muller d'ordre 2.

La troisième partie concerne l'application directe de ces algorithmes de décodage sur le DES. Nous utilisons les principes de la cryptanalyse linéaire appliqués par Matsui en 1994 et du décodage pour obtenir des propriétés statistiques sur le DES. De nombreuses expériences sont menées.

La dernière partie est une digression consacrée à l'étude des algorithmes de construction des courbes modulaires. Cette partie est davantage un "survey" des idées développées par G. Frey et M. Muller. Notre but ici est de construire des courbes modulaires et de calculer le cardinal de leur jacobienne.

Mots clefs

Testeurs, complexité, algorithmes de décodage, code de Reed-Muller, chiffrement par blocs, cryptanalyse linéaire, DES, courbes modulaires.

Thèse préparée à l'INRIA Rocquencourt dans le projet CODES

Imprimé à l'INRIA