D.E.A de mathématiques et d'informatique de Bamako

Année 2007

Cours de Codes Correcteurs d'Erreurs (et fonctions boolénnes)

Claude Carlet

Table des matières

Bibliographie

[1] *Algèbre discrète et codes correcteurs*, O. Papini et J. Wolfmann, Mathématiques et Applications, Collection de la SMAI, Springer-Verlag (1995).

[2] *Exercices sur les Codes Correcteurs*, A. Poli, Collection logique mathématique et informatique, Masson.

[3] *The Theory of Error Correcting Codes*, F.J. Macwilliams, N.J.A. Sloane, North-Holland 1986 (dernière édition).

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

[5] *Introduction aux méthodes de la cryptographie*, Brassart, Beckett

[6] *Boolean Functions for Cryptography and Error Correcting Codes.* C. Carlet. Chapter of the monography *Boolean Methods and Models*, Y. Crama and P. Hammer eds, Cambridge University Press, to appear. Preliminary version available at http://www-rocq.inria.fr/codes/Claude.Carlet/pubs.html

[7] *Vectorial (multi-output) Boolean Functions for Cryptography.* C. Carlet. Chapter of the monography *Boolean Methods and Models*, Y. Crama and P. Hammer eds, Cambridge University Press, to appear soon. Preliminary version available at http://www-rocq.inria.fr/codes/Claude.Carlet/pubs.html

[8] *Handbook of applied cryptography*, Menesez, Van Oorschott. Disponible sur le web à l'URL http://www.cacr.math.uwaterloo.ca/hac

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

[10] *The codebreakers*, David Kahn

Chapitre 1

Introduction

Le *codage correcteur d'erreurs*, dont l'origine remonte à la fin des années 40, permet de transmettre de façon fiable de l'information, codée au moyen de mots binaires d'une longueur donnée, sur des lignes plus ou moins bruitées. La transmission de l'information binaire sur des lignes bruitées présentant un risque d'erreurs variable selon les cas, il s'agit de trouver un moyen de les corriger à la réception de l'information, au prix d'une certaine redondance, tout en minimisant dans chaque situation le temps d'occupation de la ligne. Les premiers travaux sur le sujet ont été menés par Golay, Hamming et Shannon [1].

Les codes correcteurs d'erreurs sont présents aujourd'hui dans tous les réseaux, à des niveaux techniques plus ou moins complexes. La généralisation de l'usage des satellites de télécommunication dans les réseaux mondiaux augmentant le niveau de bruit, le niveau technique de la correction d'erreurs dans ces réseaux a tendance à augmenter sensiblement.
On trouve aussi de la correction d'erreurs à un niveau sophistiqué dans les sondes spatiales, les systèmes de guidage, les lecteurs de disques numériques et de disques compacts.

Voyons schématiquement le problème posé par la **transmission sur un canal bruité**.

`Source` $\longrightarrow$ $(u_1,\ldots,u_k)$ $\rightarrow$ `Encodeur` $\rightarrow$ $(x_1,\ldots,x_n)$

1. voir son texte fondateur de le théorie de l'information: *A mathematical theory of communication*

$$\rightarrow \quad \texttt{Canal bruité} \quad \rightarrow$$

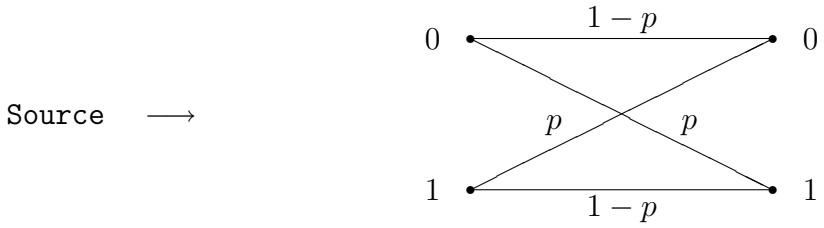
$$(y_1,\ldots,y_n) \quad \rightarrow \quad \texttt{Décodeur} \quad \rightarrow \quad (v_1,\ldots,v_k)$$

L'encodeur transforme le mot transmis en un **mot de code**.
Nous appelons **taux de transmission** le rapport $= k/n$.
La valeur r telle que $n = k + r$ est appelée **redondance**.
Dans le canal bruité, certains symboles peuvent être modifiés.
Pouvoir **détecter** une erreur, c'est pouvoir répondre à la question suivante : le vecteur reçu $(y_1,\ldots,y_n)$ est-il égal au vecteur $(x_1,\ldots,x_n)$?
Pouvoir **corriger** cette erreur, c'est pouvoir, après détection, obtenir par décodage:

$$(v_1,\ldots,v_k) = (u_1,\ldots,u_k) \ .$$

Le **modèle** utilisé pour étudier les performances des codes est

le CANAL BINAIRE SYMETRIQUE



1.1 Introduction au domaine des applications

1.1.1 La Protection de la mémoire des ordinateurs

- la **source** est une suite de données numériques.
- la **ligne** est la mémoire centrale de l'ordinateur. Elle est composée de **puces**:

$$\textbf{une puce} = 64K = 1024 * 64 = 65536 \ \text{ sites binaires.}$$

– des erreurs transitoires apparaissent dues à des particules α qui viennent modifier l'état des sites binaires.

Sans code-correcteur d'erreurs, la durée moyenne de bon fonctionnement d'un ordinateur de **256** *megaoctets serait de* **4** *heures.*

1.1.2 Les photos de la Planète MARS transmises par le vaisseau spatial MARINER 9 en 1972

La qualité de la transmission était améliorée par le **code de** Reed-Muller **d'ordre 1**.
Le code comporte 64 mots de 32 bits chacun. Il peut corriger 7 erreurs. L'encodage est très rapide; le décodage nécessite au plus 160 additions (pour chaque mot).

Le **Schéma de transmission** est le suivant:

1. Du blanc au noir, 64 teintes de gris sont définies.
2. Chaque mot de code est identifié à une teinte de gris.
3. A chaque point de la photo, est affecté une teinte de gris, et donc, par encodage, un mot P de 32 bits.

Une `ERREUR` perturbant au plus 7 bits durant la `TRANSMISSION` de P, est corrigée.

Chapitre 2

Les codes en blocs

On dit *codage par bloc*, par opposition au codage convolutif qui "traite les symboles l'un après l'autre". Ici les positions sont regroupées en blocs de même longueur.

2.1 Trois exemples historiques

2.1.1 Le test de parité: Un exemple de code détecteur.

Chaque mot de code a un nombre **pair** de bits "1" et il y a un seul bit de redondance:

$$x_n = \sum_{i=0}^{n-1} x_i \mod 2 \quad .$$

Ce code détecte un nombre impair d'erreurs. Ainsi pour $n = 6$:

$$1\ 0\ 1\ 0\ 1\ 1 \quad \longrightarrow \quad \begin{cases} 1\ 0\ 0\ 0\ 1\ 1 & \text{l'erreur est détectée} \\ 1\ 1\ 0\ 0\ 1\ 1 & \text{les erreurs ne sont pas pas détectées} \end{cases}$$

Ce code n'est pas correcteur car il ne permet pas de localiser les erreurs.

2.1.2 Le code à répétition

Dans l'encodage, chaque bit du mot-source est répété trois fois (ce qui triple la longueur; donc le taux de transmission est de $1/3 = 0.33$). Ce code peut corriger une erreur par décodage majoritaire: chaque groupe de trois bits consécutifs du mot de code est décodé en un 0 (resp. un 1) s'il contient une majorité de 0 (resp. de 1).

2.1.3 Le premier code du mathématicien R.W. HAMMING

Ce code, à l'origine "un bricolage", est basé sur le test de parité :

Longueur : $n = (t+1)^2$, Information : $k = t^2$, Redondance : $r = 2t+1$.

Son fonctionnement est clair dès que l'on regarde un exemple. Ainsi pour $t = 2$, et donc $n = 9$, $k = 4$, $r = 5$. Il s'agit d'un code binaire de longueur 9, corrigeant 1 erreur et en détectant 3.

ENCODAGE

$$1\ 1\ 0\ 1 \quad \longrightarrow \quad \begin{array}{cc|c} 1 & 1 & 0 \\ 0 & 1 & 1 \\ \hline 1 & 0 & 1 \end{array} \quad \longrightarrow \quad 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1$$

(chaque ligne et chaque colonne ont alors un nombre pair de "1")

DECODAGE

$$1\ 1\ 0\ 0\ \underline{0}\ 1\ 1\ 0\ 1 \quad \longrightarrow \quad \begin{array}{cc|c} 1 & 1 & 0 \\ 0 & \underline{0} & 1 \\ \hline 1 & 0 & 1 \end{array} \quad \longrightarrow \quad 1\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1$$

Le taux de transmission est passé de $1/3 = 0.33$, qui était le taux de transmission du code à répétition 1-correcteur, à $4/9 = 0.44$.

Le deuxième code de HAMMING, appelé définitivement *code de Hamming*, aura un taux de $4/7 = 0.55$, le meilleur possible pour un code 1-correcteur transmettant des blocs de 4 bits.

2.2 Généralités

Le codage est dit *systématique* quand les blocs de contrôle sont regroupés à la fin du mot émis (le mot M qui va transiter sur la ligne):

$$\boxed{m = (x_1, \ldots, x_k)} \quad \rightarrow \quad \texttt{encodage} \quad \rightarrow \quad \boxed{M = (x_1, \ldots, x_k | x_{k+1}, \ldots, x_{k+r})}$$

Le vecteur M est un *mot de code*; *le code* est l'ensemble des mots de code, un ensemble de messages particuliers parmi les messages de longueur $n = k + r$.
Dans l'ensemble des messages de longueur n, on définit une métrique.

Définition 1 *Soit A un alphabet ; $a \in A^n$ et $b \in A^n$. La* **Distance de Hamming** *du mot a au mot b est;*

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

Cette distance vérifie les propriétés usuelles des distances:

$$\begin{array}{rcl} \text{Symétrie} & : & d(a,b) = d(b,a) \\ \text{Positivité} & : & d(a,b) \geq 0 \\ & & d(a,b) = 0 \quad \Leftrightarrow \quad a = b \\ \text{Inégalité triangulaire} & : & d(a,b) \leq d(a,c) + d(c,b) \end{array}$$

C'est la distance qui détermine la capacité de correction d'un code:

Soit C un code. On définit ainsi la **distance minimale** *d de C:*

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

Alors le code C est $(d-1)$-détecteur d'erreurs et $[\frac{d-1}{2}]$-correcteur d'erreurs.

En effet, le code est t-détecteur si et seulement si toute boule centrée en un mot de code x et de rayon t ne rencontre aucun autre mot de code (rappelons qu'en notant A l'alphabet et n la longueur, une telle boule est l'ensemble: $B(x,t) = \{y \in A^n \mid d(x,y) \leq t\}$). La plus grande valeur possible pour t est donc $d - 1$.
Le code est t-correcteur si et seulement si deux boules de rayon t et centrées en des mots de code différents ne se rencontrent jamais.
Exercice 1 *Montrer, en utilisant la propriété d'inégalité triangulaire de la distance de Hamming, que la condition "deux boules de rayon t et centrées en des mots de code différents ne se rencontrent jamais" équivaut à $t \leq \left[\frac{d-1}{2}\right]$.*

Il existe des relations entre les paramètres d'un code. Ainsi **l'Inégalité de Hamming**:

Soit C un code binaire de longueur n, cardinal 2^k et distance minimale d. Soit $r = n - k$ la redondance. Le code C est e-correcteur avec $e = [(d-1)/2]$.

Alors n, e et r sont liés par :

$$1 + \left(\begin{array}{c} n \\ 1 \end{array} \right) + \ldots + \left(\begin{array}{c} n \\ e \end{array} \right) \leq 2^r$$

(le terme de gauche représente le nombre des éléments d'une boule de centre $c \in C$ et rayon e; or ces boules sont disjointes et leur nombre est 2^k).

`Exemple :` $A = \{0,1\}$, $k = 4$, $e = 1 \implies r = 3$.
Le meilleur taux de transmission pour ces hypothèses est $4/7$. Il est obtenu avec le code de HAMMING (7,4,3) binaire.

Définition 2 *Soit C un code de longueur n, e-correcteur. Le code C est dit* **parfait** *si et seulement si:*

$$F_2{}^n = \bigcup_{c \in C} \{\ x \in F_2{}^n \mid d(c,x) \leq e\ \}$$

Un code est donc *parfait* si l'espace ambiant est la réunion des boules de rayon e centrées en un mot de C, c'est à dire si l'inégalité de Hamming est une égalité.

Remarque 1 *Il y a peu de codes parfaits, essentiellement : les codes de* HAMMING *et de* GOLAY.

Chapitre 3

Codes linéaires

Etant donné un code C de cardinal M, pour calculer la distance minimale de C, il faut à priori faire $\begin{pmatrix} M \\ 2 \end{pmatrix} = \frac{M(M-1)}{2}$ calculs de distances. Si n est la longueur du code, ce calcul est de complexité $\mathcal{O}(n\,M^2)$. On a donc cherché à construire des codes pour lesquels ce calcul était minimisé. L'idée qui s'est rapidement imposée est:
- de choisir un alphabet sur lequel on dispose d'une opération commutative (notée $+$) pour laquelle chaque élement u de l'alphabet admet un symétrique $-u$; la distance de Hamming entre deux mots de codes a et b est alors le poids de Hammings du mot $a - b$ (voir la définition ci-dessous).
- de choisir des codes tels que si a et b sont des mots de code, alors $a - b$ est aussi un mot de code; le calcul de la distance minimale se ramène alors à celui du poids (non nul) minimal du code; il est de complexité $\mathcal{O}(n\,M)$.
En d'autres termes, on prend pour alphabet A un groupe commutatif et pour code un sous-groupe de A^n.
Un deuxième problème se posait: celui du stockage du code. En effet, supposons qu'on ait réussi à construire un code de "bonne" distance minimale. Pour pouvoir l'utiliser, disons comme code détecteur d'erreurs, encore faut-il être capable de reconnaître si un mot reçu appartient au code. Et pour cela, mieux vaut ne pas avoir à stocker les mots de code dans un tableau, dont la taille serait irraisonnable dès que la longueur et le cardinal du code atteindraient des grandeurs réalistes. Pour résoudre ce deuxième problème:
- on choisit pour alphabet un corps fini K; cela autorise les opérations de multiplication et de division (par un élément non nul);
- on choisit pour code un sous-groupe de K^n qui soit de surcroît stable par

combinaisons linéaires; cela permet de représenter le code par sa matrice génératrice (voir ci-dessous) qui est une représentation compacte du code (si M est le cardinal et n la longueur, alors pour un code binaire, la taille de la matrice génératrice est $n\,log(M)$ alors que celle du code est $n\,M$, exercice!). Un code linéaire est donc un sous-espace vectoriel d'un espace vectoriel fini sur un corps fini (l'alphabet). Ainsi, la *théorie algébrique des codes est basée sur la théorie des corps finis et sur son algorithmique.* Elle utilise la terminologie de l'algèbre vectorielle, d'abord pour préciser les structures étudiées.

Définition 3 *Soit K un corps fini et soit $n > 0$. Le K-espace vectotiel K^n est muni de la métrique de Hamming. Un* **code linéaire** *est un K-sous-espace de K^n. Ses paramètres sont : sa* **longueur** *n, sa* **dimension**, *sa* **distance minimale**.
Ces deux derniers paramètres sont notés généralement k et d. On dit que le code C **est un code [n,k,d]**.

Le nombre de mots de codes étant $|K|^k$, on devra choisir k de sorte que ce nombre soit supérieur ou égal au nombre d'informations distinctes qu'on est susceptible de faire transiter à chaque envoi d'un mot sur la ligne.

Le **poids** d'un mot de code est le nombre de ses symboles non nuls.

$$\forall a \in C, a = (a_1, \ldots, a_n) \ : \ \omega(a) = |\ \{\ i \in [1,n] \mid a_i \neq 0\ \}\ | \ .$$

Dans le cas binaire, on a par le poids du mot un renseignement complet sur la distribution des valeurs des coordonnées du mot. Ce n'est pas le cas lorsque l'alphabet n'est pas réduit à $\{0,1\}$. Le poids est alors un élément du **poids complet**, qui donne pour chaque valeur possible des symboles le nombre de symboles ayant cette valeur.

Dans ce paragraphe, C désigne un code linéaire $[n,k,d]$ sur K. On voit clairement que

$$\begin{aligned} d &= \min_{a \in C, b \in C} \{\ \omega(a-b) \mid a \neq b\ \} = \min\ \{\ d(a-b,0) \mid a \neq b\ \} \\ &= \min_{c \in C*} \{\ \omega(c)\ \} \end{aligned}$$

Proposition 1 *La distance minimale d'un code linéaire est égale au plus petit poids non nul de ce code.*

3.1 Matrice génératrice et matrice de contrôle

Il y a deux manières de représenter un code linéaire à l'aide de matrices. Soit on introduit un homomorphisme dont le code est l'espace vectoriel image; on obtient ainsi la notion de matrice génératrice. Soit on introduit un homomorphisme dont le code est le noyau; on obtient ainsi la notion de matrice contrôle.

3.1.1 Matrice génératrice

Pour connaître le code en tant que sous-espace, il suffit d'en avoir une base. Celle-ci est le plus souvent représentée sous la forme d'une matrice $k \times n$ sur K, la **matrice génératrice** du code, dont les lignes sont les vecteurs de cette base.
Pour former un mot de code, on calcule le produit d'un vecteur-ligne $(u_1, \ldots, u_k)$ et de la matrice génératrice:

$$[u_1, \ldots, u_k] \quad \begin{bmatrix} g_1^1 & g_1^2 & \ldots & g_1^n \\ & & \ldots & \\ g_k^1 & g_k^2 & \ldots & g_k^n \end{bmatrix} \quad = \quad [x_1, \ldots, x_n]$$

Cette matrice peut avoir une forme *systématique*, qui rend le codage *lisible*:

`Codage systématique`

$$\underbrace{[x_1, \ldots, x_k]}_{\texttt{message}} \quad \underbrace{\left| \begin{matrix} 1 & 0 & \ldots & 0 & v_1^1 & \ldots & v_1^r \\ & & & & & & \\ 0 & \ldots & 0 & 1 & v_k^1 & \ldots & v_k^r \end{matrix} \right|}_{\texttt{matrice génératrice}} = \underbrace{[x_1, \ldots, x_k], x_{k+1}, \ldots, x_{k+r}]}_{\texttt{mot émis}}$$

ce qui signifie que pour tout $j = 1, \ldots, r$, on a:

$$x_{k+j} = \sum_{i=1}^{k} x_i v_i^j. \tag{3.1}$$

Exercice 2 *Donner les matrices génératrices du code de parité (cf. page 7) et du premier code de Hamming.*

Remarque 2 *Soit C un code linéaire [n,k,d]. L'encodage se fait en multipliant le mot source par la matrice génératrice du code. Le mot source doit donc être de longeur k. La redondance est de $n - k$ symboles.*

3.1.2 Code dual et matrice de contrôle

Une autre façon de définir un code linéaire (c'est à dire un sous-espace vectoriel de K^n) est de donner une application linéaire dont il est le noyau. On obtient ainsi une matrice H telle que

$$C = \{(x_1, \cdots, x_n);\ H \times \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = 0\}$$

(où 0 désigne le vecteur nul).
H est la matrice génératrice d'un autre code appelé code dual de C: on désigne par $< , >$ le produit scalaire usuel: $< x,x' >= \sum_{i=1}^{n} x_i x'_i$ (calcul effectué dans le corps).
Le **code dual** du code C est son espace orthogonal:

$$C^{\perp} = \{\ x' \in K^n \mid \forall x \in C,\ < x,x' >= 0\ \}\ .$$

Remarque 3 *D'après la théorie des espaces vectoriels*[1]*, le dual de $C^{\perp}$ est C lui-même (et cette propriété est caractéristique des codes linéaires). On en déduit:*

$$C = \{\ x' \in K^n \mid \forall x \in C^{\perp},\ < x,x' >= 0\ \}\ . \tag{3.2}$$

La **matrice de parité (ou de contrôle)** du code C est la matrice génératrice de son dual $C^{\perp}$.
Un mot appartient donc au code C si et seulement si il est orthogonal à chacun des vecteurs-ligne de la matrice de parité.
$C^{\perp}$ est un code linéaire $[n,n-k,?]$ (à priori, il n'y a pas de relation simple entre d et la distance minimale de $C^{\perp}$).

Exercice 3 *Comment obtient-on la matrice génératrice de la somme directe de deux codes linéaires, à partir des matrices génératrices de ces codes?*

---

1. Une autre façon, plus algorithmique, de voir les choses est la suivante: pour qu'un mot appartienne à $C^{\perp}$, il suffit qu'il soit orthogonal à toutes les lignes de la matrice génératrice, puisque celles-ci constituent une base de l'espace vectoriel C.
$C^{\perp}$ est donc l'ensemble des solutions $x' = (x'_1, \cdots, x'_n)$ du système d'équations dans K suivant:

$$\begin{cases} g_1^1\, x'_1 + \cdots + g_1^n\, x'_n &= 0 \\ \cdots & \cdots\ \cdots \\ g_k^1\, x'_1 + \cdots + g_k^n\, x'_n &= 0 \end{cases}$$

On en déduit (voir les résultats bien connus sur l'algorithme du pivot de Gauss) que $C^{\perp}$ est un code linéaire de longueur n et de dimension $n-k$. Le dual de $C^{\perp}$ étant inclus dans C et de même dimension, il lui est égal.

En déduire la façon dont on obtient la matrice de parité de l'intersection de deux codes linéaires dont les duaux sont en somme directe.
Soit C un code linéaire binaire contenant des mots de poids impair et C' l'ensemble de ses mots de poids pair. Donner la matrice de parité de C' en fonction de celle de C.
Dans le cas d'un codage systématique, la matrice de parité est obtenue aisément.

Proposition 2 *Soit G la matrice génératrice de C, pour un codage systématique:*

$$G \ = \ [\ I_k\ ,\ M\ ]$$

où I_k est la matrice unité $k \times k$, et $M =$ une matrice $k \times r$. Alors la **matrice de parité** *de C est:*

$$H \ = \ [\ {}^tM \ \mid \ -I_{n-k}\ ]$$

Preuve: Les notations sont celles de la page 13. En vertu de (3.1), si le vecteur: $[x_1,\ldots,x_k,x_{k+1},\ldots,x_{k+r}]$ appartient au code C alors pour tout $j = 1,\ldots,r$, on a $\sum_{i=1}^{k} x_i v_i^j \ - \ x_{k+j} \ = 0$, ce qui est équivalent au fait que le vecteur $[x_1,\ldots,x_k,x_{k+1},\ldots,x_{k+r}]$ est orthogonal à la $j$ième ligne de la matrice H. D'après (3.2), le code C est donc inclus dans le code de matrice de parité H. Il lui est donc égal puisqu'il a même dimension. $\diamond$

Remarque 4 *Interprétation des lignes et des colonnes des matrices génératrice et de parité:*
On a vu la signification, relativement à un code linéaire C, des lignes de la matrice génératrice G: elles engendrent par combinaisons linéaires tous les mots du code.
La signification des lignes de la matrice de parité H est claire elle aussi: d'après (3.2), un mot appartient au code C si et seulement si il est orthogonal à chacune des lignes de H.
Les colonnes de G et de H ont, elles aussi, une signification: notons $V_1,\ldots,V_n$ les colonnes de G et $W_1,\ldots,W_n$ celles de H. Alors:
- les mots du code C sont les mots de la forme $(< u,V_1 >,\ldots,< u,V_n >)$, où u varie dans K^k; notons que l'ensemble des applications $u \rightarrow < u,V >$ ($V \in K^k$) est celui de toutes les formes linéaires sur K^k. Les coordonnées des mots du code C peuvent donc être vues comme des formes linéaires sur

K^k; plus précisément, celles qui sont définies par les colonnes de la matrice G;
- un mot $(x_1, \ldots, x_n)$ appartient au code si et seulement si le vecteur $\sum_{i=1}^{n} x_i W_i$ est nul.

Remarque 5 *Tout code linéaire admet un codage systématique:*
la matrice génératrice G étant de rang k (par définition), elle admet k colonnes linéairement indépendantes. Quitte à permuter les colonnes de cette matrice (c'est à dire les coordonnées des mots de code) on peut donc supposer que G est de la forme $[M \mid N]$, où M est une matrice $k \times k$ régulière (i.e. inversible) et N une matrice $k \times (n-k)$. On ne change pas le code C en multipliant à gauche G par la matrice régulière M^{-1} (exercice!) et la nouvelle matrice génératrice que l'on obtient est systématique. Bien entendu, l'encodage n'est pas le même selon que l'on utilise la matrice G ou la matrice $M^{-1}G$, mais le code, lui, c'est à dire l'ensemble des mots susceptibles de transiter sur la ligne, ne change pas.

Avec les éléments de définition que nous venons d'introduire, on peut formaliser précisemment le fonctionnement d'un code linéaire.

Soit H la matrice de parité de C, soit x un mot du code C et un mot érroné: $y = x + \epsilon$.

Le symbole ϵ est le vecteur d'erreur, x étant le **mot émis** et y le **mot reçu.** Nous allons voir comment on peut formellement détecter ϵ et corriger y.

`DETECTION :` d étant la distance minimale, la capacité de détection de C est égale à $d-1$; on suppose donc que ϵ est de poids au plus $d-1$; la détection s'effectue ainsi

$$y \in C \iff y\,{}^tH = 0 \qquad ({}^tH \text{ est la transposée de } H)$$

où $y\,{}^tH$ est le **syndrome** de y.
En effet, on a vu qu'un mot appartient au code C si et seulement si il est orthogonal à chacune des lignes de la matrice H, et cela est bien équivalent au fait que le produit de ce mot (ce vecteur) et de la transposée de H est le vecteur nul.

Remarque 6 *Bien sûr, $y\,{}^tH = 0$ est équivalent à $H\,{}^ty = 0$.*

`CORRECTION :` la capacité de correction de C est égale à $e = (d-1)/2$ et on suppose que le mot-erreur $\epsilon = y - x$ est de poids au plus e. On calcule le syndrome de y, qui est aussi le **syndrome de l'erreur** :

$$s = y\ {}^tH = (x+\epsilon)\ {}^tH = \epsilon\ {}^tH\ .$$

On suppose avoir fait le précalcul de la table:

Mots de poids $\leq e$	syndromes
$\vdots$	$\vdots$

Le syndrome s de y est présent dans la table, puisque c'est celui de ϵ. De plus, il ne s'y trouve qu'une seule fois: supposons qu'il existe deux mots ϵ et ϵ' de poids au plus e qui aient le même syndrome, alors $\epsilon\ {}^tH = \epsilon'\ {}^tH$ implique $(\epsilon - \epsilon')\ {}^tH = 0$, i.e. $\epsilon - \epsilon' \in C$. Et $\epsilon - \epsilon'$ étant de poids au plus $2e < d$ en vertu de l'inégalité triangulaire, cela implique $\epsilon - \epsilon' = 0$. On obtient donc ϵ en repérant l'unique ligne de la table où l'on touve s en colonne de droite.

Exercice 4 *Tester le décodage par table du premier code de Hamming de longueur 9.*

Ce procédé de décodage est inutilisable en pratique, même pour des codes binaires, sauf pour de petites longueurs et de faibles capacités de correction: le nombre de mots-erreurs sur un corps d'ordre q est $1+(q-1)\begin{pmatrix} n \\ 1 \end{pmatrix} + \ldots + (q-1)^e \begin{pmatrix} n \\ e \end{pmatrix}$. Elle est toutefois la référence de développements combinatoires en vue d'obtenir ou de caractériser des objets de structure particulière. Ainsi *tout mot de l'espace ambiant peut être décodé par rapport à un code parfait.*

EXEMPLE: **Le code de Hamming binaire**
Pour chaque $m > 1$, nous notons $\mathcal{H}_m$ le code de Hamming binaire. Une façon de définir ce code est de se donner sa matrice de parité H_m: c'est la matrice $m \times (2^m - 1)$ dont les colonnes sont les éléments non nuls de l'espace $\{0,1\}^m$.

Exercice 5 *Montrer que la distance minimale de $\mathcal{H}_m$ est 3. Montrer plus généralement que la distance minimale d'un code linéaire est égale au plus petit nombre possible de colonnes de sa matrice de parité qui soient linéairement dépendantes.*

Donc $\mathcal{H}_m$ est un code linéaire $[\ 2^m-1\ ,\ 2^m-m-1\ ,\ 3\ ]$. Il est dit **primitif** (un code sur $\{0,1\}$ est dit primitif s'il est de longueur 2^m ou 2^m-1). On montre que $\mathcal{H}_m$ est un code 1-correcteur parfait :

$$n = 2^m - 1\ ,\ k = n - m \quad \Longrightarrow \quad 2^n = 2^k(2^m) = 2^k(n+1);$$

or il y a exactement $n+1$ mots dans chaque boule de rayon 1 centrée en un mot du code. Sous sa forme systématique, sa matrice génératrice est obtenue ainsi:

$$H_m = [\ M\ ,\ I_m\ ] \quad \Rightarrow \quad G_m = [\ I_{2^m-m-1}\ ,\ {}^tM\ ]\ .$$

Ainsi, par exemple, lorsque $m = 3$, $\mathcal{H}_3$ est [7,4,3], et l'on a:

$$H_3 = \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \qquad G_3 = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

Exercice 6 *Décoder le mot reçu* (0111110) *pour le code de Hamming* $\mathcal{H}_3$.

3.2 Les codes "Maximum Distance Separable" (MDS)

Soit C un code linéaire $[n,k,d]$ (i.e. de longueur n, de dimension k et de distance minimale d) sur un corps fini K. Soient H sa matrice de parité et G sa matrice génératrice. Alors le *rang* de H est égal à $n-k$. Donc il y a au plus $n-k$ colonnes dans H linéairement indépendantes. Il existe donc dans C des mots de poids $n-k+1$. On obtient :

$$d \leq n-k+1 \quad .$$

Cette relation entre les paramètres du code C est la **borne de** SINGLETON.

Remarque: Soit C un code non nécessairement linéaire sur un alphabet K de cardinal q, n la longueur du code C, $M = q^k$ son cardinal et d sa distance minimale. Soient deux mots x et y obtenus en supprimant les $d-1$ dernières coordonnées de deux mots de code différents. x et y ont au moins une coordonnée différente. Donc a $x \neq y$.
On en déduit que le code obtenu à partir de C en éliminant les $d-1$ dernières coordonnées de chaque mot de code est de cardinal M, que $M \leq q^{n-d+1}$ et donc que $k \leq n-d+1$ (borne de Singleton pour les codes quelconques).
Cette borne exprime que la distance minimale d'un code linéaire est bornée supérieurement, et nous amène tout naturellement à définir les codes linéaires dont la distance minimale est maximale:

Définition 4 *Le code C est dit "maximum distance separable" – i.e. est un code MDS – si et seulement si $d = n-k+1$ – i.e. sa matrice de parité est de rang $d-1$ (ou encore sa matrice génératrice est de rang $n-d+1$). Il est dit MDS trivial lorsque $k=1$ ou $k \geq n-1$.*

`Exemple :` On peut aisément construire un code MDS trivial binaire de type $[n,n-1,2]$. Nous donnons ci-après un exemple de matrice génératrice pour $n=6$; la généralisation, pour toute longueur, est évidente. Le code de matrice génératrice

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

est MDS trivial binaire.

Citons les propriétés immédiates de ces codes:

1. C est MDS si et seulement si chaque ensemble de $n-k$ colonnes de sa matrice de parité H est de rang $n-k$;
   Preuve: la propriété est nécessaire puisque, d étant la distance minimale de C, chaque ensemble de $d-1$ colonnes de la matrice de parité est de rang $d-1$; elle est suffisante car elle implique que la distance minimale est au moins $n-k+1$.
2. Si C est MDS, alors $C^{\perp}$ l'est aussi;
   Preuve: H est la matrice génératrice de $C^{\perp}$; $n-k$ colonnes quelconques de H sont linéairement indépendantes et constituent donc une matrice $(n-k)\times(n-k)$ régulière; les lignes d'une telle matrice étant donc linéairement indépendantes, un mot non nul de $C^{\perp}$ ne peut avoir $n-k$ coordonnées nulles; la distance minimale de $C^{\perp}$ est donc au moins $n-(n-k)+1=k+1$.
3. C est MDS si et seulement si chaque ensemble de k colonnes de sa matrice génératrice G est de rang k.

On a certaines informations sur les codes MDS; ainsi ils constituent une classe de codes dont on connait la distribution des poids

3.3 Polynômes des poids des codes linéaires

La distribution des poids d'un code est le plus souvent représentée par un polynôme à deux indéterminées. La raison de ce choix est la relation de Mac Williams ci-dessous, qui est un résultat clé de la théorie des codes.

Définition 5 *Le* **polynôme des poids** *d'un code C, linéaire ou non linéaire, est:*

$$W_C(X,Y) = \sum_{i=0}^{n} A_i \; X^{n-i} \; Y^i \quad , \; où \quad A_i = |\; \{\; u \in C \mid \omega(u) = i \;\}\; |.$$

Le théorème de Mac Williams est le suivant:

Théorème 1 *Soit C est un code linéaire de longueur n sur K, où K désigne un corps fini. Alors les polynômes des poids de C et de son dual $C^{\perp}$ sont liés par la relation suivante (dite relation de Mac Williams):*

$$K = GF(2) \;:\; W_{C^{\perp}}\;(X,Y) = \frac{1}{|C|}\; W_C\;(X+Y, X-Y) \quad \textit{cas binaire}\;.$$

$K = GF(q)$: $W_{C^\perp}(X,Y) = \frac{1}{|C|} W_C(X + (q-1)Y, X - Y)$ *cas général .*

où $GF(q)$ est le corps fini à q éléments (on le note $GF(q)$ car les anglo-saxons l'appellent "Galois field").

EXEMPLE : $K = \{0,1\}$, $C := \{ 0\,0\,0\,0 , 1\,1\,1\,1 \}$.

$$W_C(X,Y) = X^4 + Y^4 \quad , \quad C \text{ est un code [4,1,4].}$$
$$W_{C^\perp}(X,Y) = \frac{1}{2} [(X+Y)^4 + (X-Y)^4]$$

$$W_{C^\perp}(X,Y) = X^4 + 6X^2Y^2 + Y^4 \quad , \quad C \perp \text{ est un code [4,3,2].}$$

Pour simplifier, nous ne prouvons le Théorème 1 que dans le cas binaire. La preuve se généralise assez facilement. Nous utiliserons la transformée de *Fourier discrète* (ou de *Hadamard* ou encore de *Walsh*) et le lemme suivant:

Lemme 1 *Soit une application $f : K^n \longmapsto \mathbf{R}$ ($K = GF(2) = \{0,1\}$) et soit:*

$$\widehat{f} : K^n \mapsto \mathbf{R} , \quad \widehat{f}(u) = \sum_{v \in K^n} (-1)^{<u,v>} f(v) \quad ,$$

la transformée de FOURIER *de f. Alors :*

$$\sum_{u \in C^\perp} f(u) = \frac{1}{|C|} \sum_{u \in C} \widehat{f}(u) \quad .$$

Remarque 7 *Dans la somme $\sum_{v \in K^n} (-1)^{<u,v>} f(v)$, on peut considérer que le produit scalaire $< u,v >= \sum_{i=1}^{n} u_i v_i$, au lieu d'être calculé dans K, l'est dans $\mathbf{Z}$ (puisqu'il est en exposant de (-1), dont le carré est 1).*

Preuve du lemme:

$$\frac{1}{|C|} \sum_{u \in C} \widehat{f}(u) = \frac{1}{|C|} \sum_{v \in K^n} f(v) \sum_{u \in C} (-1)^{<u,v>} \quad .$$

$v \in C^\perp \quad \Longrightarrow \quad \sum_{u \in C} (-1)^{<u,v>} = \mid C \mid$

$v \notin C^{\perp} \quad \Longrightarrow \quad \exists u_0 \in C \,|\, < u,v > \neq 0 \quad \Longrightarrow \quad \sum_{u\in C}(-1)^{<u,v>} = \sum_{u\in C}(-1)^{<u+u_0,v>} = (-1)^{<u_0,v>}\sum_{u\in C}(-1)^{<u,v>} = 0$

$\diamond$

Preuve du Théorème : On définit l'application

$$f(u) = X^{n-\omega(u)}Y^{\omega(u)} \quad , \; u \in K^n \quad .$$

$\Longrightarrow$ on obtient le polynôme des poids de $C^{\perp}$:

$$\sum_{u\in C^{\perp}} f(u) = \sum_{u\in C^{\perp}} X^{n-\omega(u)}Y^{\omega(u)} = W_{C^{\perp}}(X,Y) \quad .$$

On applique le Lemme $\Rightarrow$

$$\begin{aligned} W_{C^{\perp}}(X,Y) &= \frac{1}{|C|}\sum_{u\in C}\widehat{f}(u) = \frac{1}{|C|}\sum_{u\in C}\sum_{v\in K^n}(-1)^{<u,v>}f(v) \\ &= \frac{1}{|C|}\sum_{u\in C}\sum_{(v_1,\ldots,v_n)}\prod_{i=1}^{n}(-1)^{u_i v_i}X^{1-v_i}Y^{v_i} \end{aligned}$$

car $< u,v >= \sum_i u_i v_i$ et $\omega(v) = \sum_i v_i$. On obtient:

$$W_{C^{\perp}}(X,Y) = \frac{1}{|C|}\sum_{u\in C}\sum_{(v_1,\ldots,v_n)}\prod_{i=1}^{n}X^{1-v_i}((-1)^{u_i}Y)^{v_i} \quad .$$

Or, il est facile de vérifier que tout produit de la forme:

$$\prod_{i=1}^{n}(\lambda_i + \mu_i)$$

se développe en:

$$\sum_{(v_1,\ldots,v_n)}\prod_{i=1}^{n}{\lambda_i}^{1-v_i}\,{\mu_i}^{v_i} \quad .$$

Cette égalité appliquée à $\lambda_i = X$ et $\mu_i = (-1)^{u_i}Y$ donne:

$$W_{C^{\perp}}(X,Y) = \frac{1}{|C|}\sum_{u\in C}(X+Y)^{n-\omega(u)}(X-Y)^{\omega(u)} = \frac{1}{|C|}W_C(X+Y,X-Y) \quad .$$

$\diamond$

`Exemple`: *Polynôme de poids des codes de Hamming binaires.* L'alphabet est $K = \{0,1\}$. Les paramètres sont:

$$n = 2^m - 1 \ \ , \ \ k = 2^m - 1 - m \ \ , \ \ d = 3 \ .$$

En utilisant la remarque 4, on montre facilement que tous les mots non nuls du dual (appelé code simplexe) ont pour poids 2^{m-1}. En effet, les mots du code C sont les mots de la forme $(< u,V_1 > , \ldots , < u,V_n >)$, où u varie dans F_2^m et où $\{V_1, \ldots ,V_n\}$ est l'ensemble des mots non nuls de longueur m, et l'application $V \mapsto < u,V >$ est linéaire. On obtient:

$$W_{\mathcal{H}_m^\perp}(X,Y) = X^n + n \ X^{(n-1)/2} \ Y^{(n+1)/2} \ .$$

On calcule alors le polynôme des poids de $\mathcal{H}_m$, par la transformée de Mac-Williams :

$$W_{\mathcal{H}_m}(X,Y) = \frac{1}{n+1} \ [\ (X+Y)^n + n \ (X+Y)^{(n-1)/2} \ (X-Y)^{(n+1)/2} \ ].$$

Voyons un exemple en petite longueur, $\mathbf{n = 31}$:

$$\begin{aligned} W(1,z) &= 1 + 155z^3 + 1085z^4 + 5208z^5 + 22568z^6 + 82615z^7 + 247845z^8 \\ &+ 628680z^9 + 1383096z^{10} + 2648919z^{11} + 4414865z^{12} \\ &+ 6440560z^{13} + 8280720z^{14} + 9398115z^{15} + 9398115z^{16} \\ &+ 8280720z^{17} + 6440560z^{18} + 4414865z^{19} + 2648919z^{20} \\ &+ 1383096z^{21} + 628680z^{22} + 247845z^{23} + 82615z^{24} \\ &+ 22568z^{25} + 5208z^{26} + 1085z^{27} + 155z^{28} + z^{31} \end{aligned}$$

On en déduit que **le code $\mathcal{H}_5$ a tous les poids sauf 1, 2, n-1, n-2**.

3.4 Décodage des codes linéaires

On a vu ci-dessus que le décodage par table est impossible à réaliser en pratique car la taille de la table est en général trop élevée. Et si on ne fait pas le précalcul de la table (i.e. si on calcule la table au fur et à mesure) c'est le temps de calcul qui pose problème. Chaque classe de codes linéaires a donc son algorithme de décodage efficace propre pour pouvoir être utilisée en pratique. Nous indiquons ci-dessous deux principes généraux de ces algorithmes dans le cas des codes binaires (i.e. sur F_2).

3.4.1 Décodage par décision majoritaire

On appelle équation de contrôle toute équation de la forme $< L_i,x >= 0$ satisfaite par tous les mots de code (i.e. telle que L_i est combinaison linéaire des lignes de la matrice de contrôle).

Définition 6 *Un système (S) d'équations de contrôle est dit orthogonal par rapport à un sous ensemble P de $\{1,\cdots,n\}$ si*

1. *Pour chaque $i \in P$, le terme x_i apparaît dans chaque équation de (S) avec un coefficient non nul;*
2. *Pour chaque $i \notin P$, le terme x_i apparaît au plus une fois dans une équation de (S) avec un coefficient non nul.*

Si $P = \{k\}$, on dit que (S) est orthogonal par rapport à la position k.

Exercice 7 *Soit (S) un système de ℓ équations de contrôle $< L_j,x >= 0$, $j = 1,\cdots,\ell$, orthogonal par rapport à la position k. Soit y le mot reçu et ϵ le mot-erreur, supposé de poids $\leq \ell/2$.*
Soient les ensembles $J_0 = \{j = 1,\cdots,\ell;\ < L_j,y >= 0\}$ et $J_1 = \{j = 1,\cdots,\ell;\ < L_j,y >= 1\}$. Montrer que si $card(J_1) > card(J_0)$ alors $\epsilon_k = 1$ et sinon $\epsilon_k = 0$.

Le principe du décodage est le suivant: on suppose que pour chaque entier k on dispose d'un système d'équations de contrôle orthogonal par rapport à la position k. On applique alors le décodage majoritaire déduit de l'exercice ci-dessus. On pourra corriger jusqu'à $e = \left[\frac{d-1}{2}\right]$ erreurs si, pour tout $k =$

$1,\ldots,n$, le nombre ℓ de l'exercice ci-dessus est au moins égal à $d-1$.

Exercice 8 *Appliquer ce décodage au code simplexe* [15,4,8] *de matrice génératrice*

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

et au mot reçu $y = 111000000000000$.
Nota Bene: pour chaque paire de colonnes de G, il existe une colonne de G qui en est la somme; cela mène à une équation de contrôle de poids 3.

3.4.2 Décodage par permutations

Exercice 9 *Soit C un code* [n,k] *systématique et e-correcteur. Soit un mot reçu y et le mot-erreur correspondant* ϵ *de poids* $\leq e$. *Montrer que le syndrome s de y est de poids au plus e si et seulement si les k premiers symboles de y sont corrects.*
Nota Bene: on montrera que le mot y et le concaténé du mot nul $(0,\cdots,0)$ *et de s ont même syndrome.*

Définition 7 *On appelle ensemble de permutations de décodage pour C tout ensemble P de permutations de* $\{1,\cdots,n\}$ *qui laisse le code globalement invariant et tel que pour tout ensemble* $E \subset \{1,\cdots,n\}$ *de cardinal au plus e, il existe une permutation dans P qui envoie E dans* $\{k+1,\cdots,n\}$.

Le principe du décodage est de trouver un ensemble de permutations de décodage pour C de cardinal aussi petit que possible et de chercher une permutation dans P telle que le syndrome du permuté de y soit de poids au plus e. Les k premières coordonnées de ce permuté sont alors correctes. On retrouve alors le mot permuté de x complet en utilisant la matrice génératrice G.

Exercice 10 *Soit le code de Golay de matrice génératrice* $G = [Id|M]$ *avec*

$$M = \begin{bmatrix} Id_3 & A & A^2 & A^4 \\ A & Id_3 & A^4 & A^2 \\ A^2 & A^4 & Id_3 & A \\ A^4 & A^2 & A & Id_3 \end{bmatrix}$$

et

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Décoder le mot $y = 000000000000100001110001$. *On admettra qu'un ensemble minimal de permutations de décodage pour ce code est de la forme* $t^i \circ s^j$, $i \in \{0,1\}$, $j \in \{0,\cdots,6\}$ *où*

$$s = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 1 & 2 & 3 & 7 & 8 & 9 & 16 & 17 & 18 & 22 & 23 & 24 & 4 & 5 & 6 & 10 & 11 & 12 & 13 & 14 \end{pmatrix}$$

$$\begin{pmatrix} 21 & 22 & 23 & 24 \\ 15 & 19 & 20 & 21 \end{pmatrix}.$$

et

$$t = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

$$\begin{pmatrix} 21 & 22 & 23 & 24 \\ 9 & 10 & 11 & 12 \end{pmatrix}.$$

Chapitre 4

Les codes de Reed-Muller

Les codes considérés jusqu'à maintenant sont des codes de grande cardinalité et de (relativement) faible capacité de correction. Cherchons maintenant à construire des codes de forte capacité de correction (dont les cardinalités seront bien sûr, à priori, plus faibles). Nous connaissons déjà un exemple de tel code: le dual du code de Hamming de longueur $n = 2^m - 1$, dont la matrice génératrice est $\mathcal{H}_m$. On a vu qu'il a un seul poids non nul: 2^{m-1}. Il peut donc corriger jusqu'à 25% d'erreurs. Son cardinal est seulement $n + 1$. On peut doubler ce cardinal, sans changer la distance minimum, en n'augmentant la longueur que de 1: ajoutons à chaque mot du code un symbole de parité; cela revient à ajouter une colonne de 0 à la matrice génératrice. Ajoutons maintenant une ligne supplémentaire constituée de 1 à cette matrice.

Exercice 11 *Montrer que le code de matrice génératrice ainsi définie a tous ses mots de poids* 0, 2^{m-1} *ou* 2^m.

Le code qu'on obtient s'appelle le code de Reed-Muller d'ordre 1. Il a une interprétation en termes de fonctions booléennes: la i-ème ligne ($i = 1, \ldots, m$) de sa matrice génératrice est la liste des valeurs prises par la i-ème fonction coordonnée de l'espace $(F_2)^m$. Une combinaison linéaire quelconque de ces lignes est donc une forme linéaire quelconque sur cet espace. La dernière ligne peut être considérée comme la liste des valeurs prises par la fonction constante 1. Une combinaison linéaire quelconque de toutes les lignes de la matrice génératrice est donc une fonction affine quelconque sur $(F_2)^m$.

Avant de généraliser cette définition de code, nous rappelons quelques éléments de théorie des fonctions booléennes.

Définition 8 *Soit* $F = F_2$ *et* $V_m = F^m$. *On appelle* fonction booléenne *sur* V_m *toute application de* V_m *dans* F.

On munit naturellement l'ensemble des fonctions booléennes des opérations induites par les opérations du corps F.

Exercice 12 *Montrer que l'ensemble des fonctions booléennes sur V_m est un espace vectoriel de dimension 2^m sur F, une base étant constituée des indicatrices (fonctions caractéristiques) des singletons.*

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

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

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

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

Théorème 2 *L'ensemble des monômes constitue une base de l'espace vectoriel des fonctions booléennes sur V_m.*

Preuve:
Montrons d'abord que l'ensemble des monômes est une famille génératrice. En vertu de l'exercice 12, il suffit pour cela de montrer que l'indicatrice d'un singleton quelconque se décompose sur cette famille. Soit donc $a = (a_1, \ldots, a_m)$ un élément quelconque de V_m et 1_a l'indicatrice du singleton $\{a\}$; en développant l'égalité suivante:

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

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

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

Elle s'écrit:

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

où, pour tout u, le coefficient a_u est dans F.

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

A toute fonction booléenne sur V_m, on peut faire correspondre un mot binaire de longueur $n = 2^m$ de la façon suivante:
on choisit un ordre sur les mots binaires de longueur m: $V_m = \{\gamma_1, \ldots, \gamma_n\}$, et on fait correspondre à une fonction $f(x)$ le mot binaire de longueur n égal à:

$$(f(\gamma_1), \ldots, f(\gamma_n)).$$

Ce mot s'appelle le *mot-image* de la fonction f (mais ce terme n'est pas standard; certains utilisent, de façon impropre, celui de table de vérité de f).

Définition 10 *Pour tout k compris entre 0 et m, on appelle code de Reed-Muller d'ordre k et on note RM(k,m) l'ensemble des mots-image des fonctions booléennes de degrés inférieurs ou égaux à k.*
On appelle code de Reed-Muller raccourci d'ordre k et on note RM(k,m), l'ensemble des mots du code RM(k,m) privés de leur première coordonnée (en supposant $\gamma_1 = 0$).*
RM(k,m) et RM*(k,m) sont clairement des codes linéaires.
Exercice 13 *Montrer que la dimension de RM(k,m) est :*

$$\sum_{i=0}^{k} \left(\begin{array}{c} m \\ i \end{array} \right).$$

A l'avenir, nous ne ferons plus la distinction entre une fonction booléenne et son mot-image.
D'après le théorème 2, RM(m,m) est l'ensemble des fonctions booléennes tout entier. RM(0,m) est la paire constituée des deux fonctions constantes et RM(1,m) l'ensemble des *formes affines* sur V_m (i.e. des sommes d'une forme linéaire et d'une constante).

D'après le théorème 2, une matrice génératrice de RM(m,m) s'obtient de la façon suivante: à chaque monôme, on fait correspondre une ligne de la matrice génératrice, qui s'obtient en écrivant les valeurs de cette fonction monôme en $\gamma_1, \ldots, \gamma_n$ (ainsi, la première ligne est une ligne de "1"; elle correspond au monôme de degré nul).

Le code RM($m-1$,m) et les duaux des codes de Reed-Muller.

Exercice 14 *Montrer que le poids d'un monôme est pair si et seulement si ce monôme est de degré strictement inférieur à m.*
Montrer que l'application qui à un mot (ou une fonction booléenne) fait correspondre son poids modulo 2 est linéaire. En déduire qu'une fonction booléenne est de poids pair si et seulement si elle est de degré strictement inférieur à m.

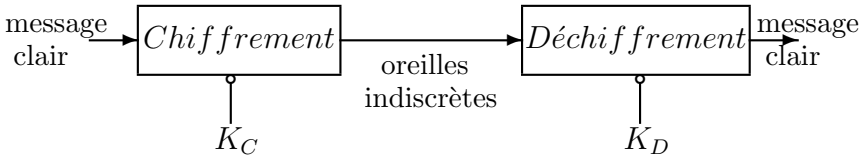
Donc, RM($m-1$,m) est l'ensemble des fonctions booléennes de poids pair.

Exercice 15 *Déduire de l'exercice précédent et de l'exercice 13 que, pour tout $r \in [0,m-1]$, le dual du code RM(r,m) est le code RM($m-r-1$,m).*

Chapitre 5

Les schémas de chiffrement par flot

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



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

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

4 - un ensemble de transformations de chiffrement (chaque transformation étant indexée par une clé):

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

5 - un ensemble de transformations de déchiffrement (chaque transformation étant indexée par une clé):

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

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

Jusqu'au milieu des année 70, les seuls cryptosystèmes connus étaient *symétriques* (on dit aussi *conventionnels* ou *à clé secrète*): la clé de chiffrement K_C était la même que la clé de déchiffrement K_D (ou du moins, la connaissance de la clé de chiffrement permettait d'en déduire la clé de déchiffrement) ce qui obligeait à garder secrète la clé K_C elle aussi.

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

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

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

- Alice choisit un cryptosystème à clé publique et utilise ce système pour envoyer secrètement à Bob un mot binaire K, qu'on appelle clé de session;

pour ce faire, elle chiffre K au moyen de la clé publique de Bob (une alternative possible est l'utilisation d'un protocole d'échange de clés tel que celui de Diffie-Hellman);
- Bob récupère K à l'aide de sa clé secrète;
- Alice et Bob détiennent maintenant un mot secret K et Alice l'utilise pour chiffrer son message long dans un cryptosystème symétrique;
- Bob peut déchiffrer le message à l'aide de K.

On considère qu'une attaque est efficace si elle a une probabilité non négigeable de réussir en un temps inférieur ou égal à quelques années (voire plus!) sur une ou plusieurs machines puissantes. Cela fixe de nos jours à 2^{80} le nombre minimal d'opérations élémentaires nécessaires à une attaque, pour qu'un cryptosystème soit considéré comme sûr. Un bon système est tel que toute attaque ait une complexité (en temps de calcul) au moins égale à celle de l'attaque exhaustive.
- *L'attaque exhaustive* est celle qui passe en revue toutes les clés possibles jusqu'à ce que le déchiffrement produise le clair. Le temps moyen de cette attaque est égal au temps d'un déchiffrement multiplié par la moitié de la taille de l'espace des clés (exercice!).

5.1 Schémas par flot

Ces schémas, qui traitent l'information bit à bit, sont très rapides. Ils peuvent être traités avec une mémoire limitée (en particulier en hardware, ce qui, permet d'optimiser leur efficacité) et la propagation des erreurs de transmission du chiffré au clair est limitée. Ils sont parfaitement adaptés à des moyens de calcul, de mémoire et de transmission limités (cryptographie en temps réel) comme la cryptographie militaire, ou la cryptographie entre le téléphone portable GSM et son réseau (chiffrement A5.1), ou dans le système de communication sans fil Bluetooth (E_0) ou dans le système de communication WLAN (RC4).

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

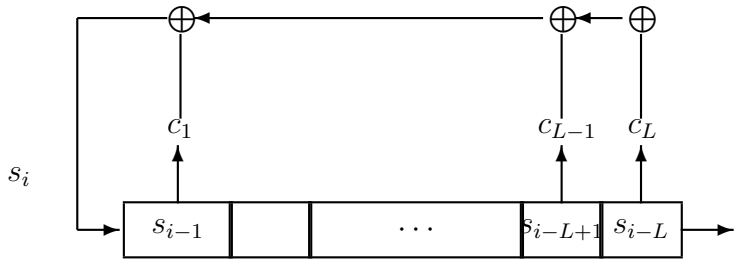
prendre d'au moins 80 bits, en pratique on prend 128 bits), qu'on appellera *clé courte*.

On a d'abord pris, historiquement, pour clé pseudo-aléatoire une suite (on dit aussi séquence) à récurrence linéaire, c'est à dire une suite satisfaisant une relation de récurrence du type:

$$s_i = \bigoplus_{j=1}^{L} c_j s_{i-j}, \forall i \geq L, \tag{5.1}$$

où les c_j sont non tous nuls.

Le procédé électronique pour générer de telles suites est le registre à décalage linéaire (*Linear Feedback Shift Register* en anglais; LFSR).



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

On se restreint au cas $c_L = 1$ (on peut s'y ramener dans le cas général à condition d'éliminer les n_0 premiers termes de la suite; prendre $c_L = 0$ est en fait déconseillé car cela donne une période plus courte). Toute suite à récurrence linéaire de ce type est périodique.

On représente classiquement la suite par sa série génératrice:

$$s(X) = \bigoplus_{i \geq 0} s_i X^i.$$

De même, on définit le polynôme de rétroaction[1] du registre:

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

La suite de série génératrice $s(X)$ est générée par le LFSR de polynôme de rétroaction $f(X)$ si et seulement si il existe un polynôme $g(X)$ de degré strictement inférieur à L tel que

$$s(X)f(X) = g(X).$$

On a

$$g(X) = \bigoplus_{i=0}^{L-1} X^i (\bigoplus_{j=0}^{i} c_{i-j} s_j). \tag{5.2}$$

On en déduit que les suites générées par un LFSR de polynôme de rétroaction $f(X)$ sont toutes les suites de séries génératrices:

$$s(X) = \frac{g(X)}{f(X)};\ d^\circ g < L.$$

Si les polynômes $f(X)$ et $g(X)$ ne sont pas premiers entre eux, alors on a $s(X) = \frac{g_0(X)}{f_0(X)}$ où $f_0(X) = \frac{f(X)}{PGCD(f(X),g(X))}$ et $g_0(X) = \frac{g(X)}{PGCD(f(X),g(X))}$. Il existe donc un LFSR plus court qui génère la même suite! Réciproquement, s'il existe un registre plus court qui génère la même suite, alors $f(X)$ et $g(X)$ ne sont pas premiers entre eux.
On appelle complexité linéaire d'une suite la longueur du plus court LFSR générant cette suite.
D'après ce qui précède, la complexité linéaire d'une suite de série génératrice $s(X) = \frac{g(X)}{f(X)}$ est égale au degré du polynôme $f_0(X) = \frac{f(X)}{PGCD(f(X),g(X))}$ (appelé polynôme minimal de rétroaction). De plus, si l'état initial du LFSR est non nul, alors la plus petite période de la suite qu'il génère est égal à l'ordre de $f_0(X)$, c'est à dire au plus petit entier n tel que $f_0(X)$ divise $X^n + 1$.

Exercice 16 *1. Montrer que, si un polynôme $f(X)$ de degré L inférieur à n est irréductible sur F_2, alors il divise $X^n + 1$ si et seulement si chaque racine de $f(X)$ (dans une extension de F_2) est une racine nième de l'unité.*
2. Montrer que tout LFSR de polynôme de rétroaction irréductible $f(X)$ de degré L et d'état initial non nul génère une suite de plus petite période égale à

1. Son polynôme réciproque, c'est à dire le polynôme $x^L \oplus c_1 X^{L-1} \oplus \cdots \oplus c_L$ est appelé le polynôme caractéristique de la récurrence.

l'ordre de $f(X)$, que l'ordre de $f(X)$ est égal à l'ordre de l'une de ses racines et que $f(X)$ est d'ordre 2^L-1 si et seulement si une de ses racines (dans un sur-corps de F_2) est primitive (i.e. toutes ses racines sont primitives).

Une suite à récurrence linéaire $s_i = \bigoplus_{j=1}^{L} c_j s_{i-j}, \forall i \geq L$ est dite ML (maximal length) si sa plus petite période est 2^L-1, c'est à dire si son polynôme de rétroacion est d'ordre 2^L-1. On dit qu'un tel polynôme est primitif. Toutes ses racines sont des éléments primitifs du corps F_{2^L} (voir annexe). Réciproquement, si une racine d'un polynôme irréductible de degré L sur F_2 est primitive alors ce polynôme est primitif.

Rappelons que, pour tout n, il existe un polynôme primitif de degré n. Les suites ML ont des propriétés très fortes:

Exercice 17 *Soit $P(X) = \sum_{i=0}^{L} p_i X^i$ un polynôme primitif de degré L à coefficients dans F_2 et soit α une racine de $P(X)$ dans le corps F_{2^L} (α est un élément primitif de ce corps). On note tr la fonction trace de F_{2^L} dans F_2 (on rappelle que cette fonction est une forme linéaire sur F_{2^L} et que toute forme linéaire sur F_{2^L} est de la forme $tr(ax)$, $a \in F_{2^L}$).*

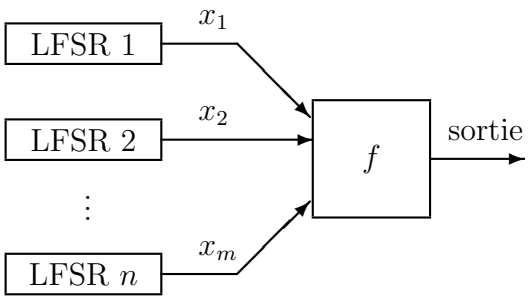
1. Montrer que toute suite de la forme $u_i = tr(a\alpha^i)$ admet $P(X)$ pour polynôme de rétroaction (c'est à dire satisfait la relation de récurrence $u_n = \sum_{i=1}^{L} p_{L-i} u_{n-i}$). En déduire que toute suite admettant $P(X)$ pour polynôme de rétroaction est de la forme $u_i = tr(a\alpha^i)$.

2. En déduire que dans toute sous-suite (i.e. toute suite constituée de termes consécutifs) de longueur $2^L + L - 2$ d'une telle suite, si on considère toutes ses sous-suites de longueur L, on trouve une et une seule fois chacune des 2^L-1 suites non nulles de longueur L sur F_2.

3. Montrer que deux suites ML correspondant au même polynôme de rétroacion $P(X)$ sont toujours les décalées l'une de l'autre et qu'étant données deux suites ML u et v quelconques de même période, il existe deux entiers positifs d (nécessairement premier avec 2^L-1) et k tels que $v_{t+k} = u_{dt}$ pour tout t.

Les schémas par flots générant la suite pseudo-aléatoire à partir d'une clé courte en utilisant un LFSR sont cassés par l'algorithme de Berlekamp-Massey: pour calculer la valeur de L et les valeurs des coefficients c_i, et pour reconstituer l'initialisation du registre, il suffit d'observer au moins $2L$ bits consécutifs de la suite pseudo-aléatoire. Si L est inconnu, il suffit donc de connaitre $2M$ bits où M est connu pour être un majorant de L. On a donc cherché d'autres moyens de générer la suite pseudo-aléatoire, en utilisant des fonctions booléennes.

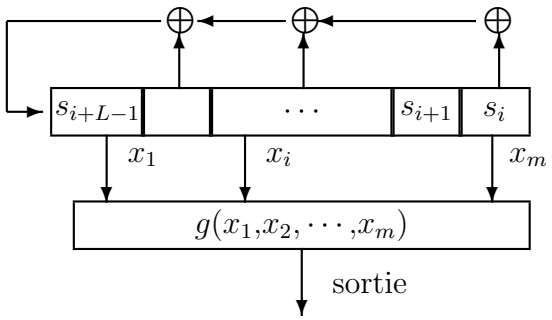
LFSR combinés (voir schéma ci-dessous)



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

- Le dernier coefficient de rétroaction de chaque LFSR est non nul (i.e. égal à 1) et le degré du polynôme de rétroacion du LFSR est donc égal à la longueur du LFSR;
- Les polynômes de rétroaction des LFSR sont primitifs, pour assurer une période maximale $2^L - 1$ de la suite produite par chaque LFSR (supposé de longeur L); ils sont donc irréductibles et la suite générée par chaque LFSR est donc de complexité linéaire égale à la longueur du LFSR;
- Les longueurs des LFSR sont premières entre elles (pour assurer que la période de la sortie multiple des LFSR est égale au produit des périodes des suites générées par les LFSR).

LFSR filtrés On a un seul LFSR et à chaque top, l'état du registre (la suite des bits qu'il contient) ou un état partiel (une sous-suite) est l'entrée d'une fonction booléenne. On montre que ce système est équivalent au précédent, mais les attaques sur ce système ne fonctionnent pas avec la même complexité que sur le système par fonction de combinaison qui lui est équivalent. Les



spécifications concernant le choix du LFSR pour un tel modèle sont les mêmes que plus haut. De plus, les positions en lesquelles les bits $x_1,x_2,\ldots$ sont prélevées ne doivent pas être régulièrement espacées.

5.2 Fonctions booléennes

5.2.1 Rappel

Définition 11 *On appelle* fonction booléenne *sur F_2^m toute application de F_2^m dans F_2.*

On munit naturellement l'ensemble des fonctions booléennes des opérations induites par les opérations du corps F_2.
L'ensemble des fonctions booléennes sur F_2^m est un espace vectoriel de dimension 2^m sur F_2, une base étant constituée des indicatrices (fonctions caractéristiques) des singletons.
Les fonctions coordonnées $x = (x_1,\ldots,x_m) \to x_i$ (que l'on notera simplement x_i) sont les fonctions booléennes les plus élémentaires, après les fonctions constantes 0 et 1.
Les produits de fonctions coordonnées sont aussi des fonctions booléennes sur F_2^m. On les appelle les *monômes*. Le degré du monôme $\prod_{i\in I} x_i$ est le cardinal $|I|$ de I. L'unique monôme de degré 0 est la fonction constante 1.

A chaque monôme correspond un mot $u \in F_2^m$ tel que ce monôme s'écrive:

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

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

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

L'ensemble des monômes constitue une base de l'espace vectoriel des fonctions booléennes sur F_2^m.

Cette écriture, unique, correspond à la représentation de la fonction booléenne comme fonction polynomiale à m variables dont le degré relatif à chacune des variables est au plus 1.

Définition 12 *L'écriture (unique) d'une fonction booléenne (resp. booléenne vectorielle) $f(x)$ comme combinaison linéaire à coefficients dans F_2 (resp. $GF(2^r)$) de monômes s'appelle la* forme algébrique normale *de $f(x)$.*
Elle s'écrit:

$$f(x) = \sum_{u \in F_2^m} a_u \left(\prod_{i=1}^{m} {x_i}^{u_i} \right) = \sum_{u \in F_2^m} a_u \, x^u$$

où, pour tout u, le coefficient a_u est dans F_2 (resp. $GF(2^r)$).

On appelle *degré* d'une fonction booléenne le degré global de sa forme algébrique normale.

5.2.2 Forme algébrique normale et transformée de Möbius

Dans la définition 12, il apparaît naturellement une deuxième fonction booléenne (resp. booléenne vectorielle) sur F_2^m: la fonction $u \to a_u$. Cette fonction s'appelle la *transformée de Möbius* de f.

On a, pour toute fonction booléenne sur F_2^m:

$$f(x) = \sum_{u \in F_2^m} g(u) \, x^u$$

où g est la transformée de Möbius de f.

Proposition 3 *On définit sur F_2^m l'ordre partiel suivant:*

$$(v_1, \ldots, v_m) \preceq (u_1, \ldots, u_m) \Longleftrightarrow \forall i = 1, \ldots, m,\ v_i \leq u_i.$$

Alors, la transformée de Möbius d'une fonction booléenne (resp. booléenne vectorielle) quelconque f est la fonction g définie par:

$$g(u) = \sum_{v \in F_2^m \,|\, v \preceq u} f(v),$$

la somme étant calculée dans F_2 (resp. $GF(2^r)$).
Preuve: Calculons d'abord l'expression de f en fonction de g: pour tout mot u et tout mot v, le produit $v^u = \prod_{i=1}^{m} {v_i}^{u_i}$ est égal à 1 si et seulement si chacun de ses facteurs vaut 1, c'est à dire si $u \preceq v$. On a donc:

$$f(v) = \sum_{u \preceq v} g(u).$$

Il nous reste à montrer que l'application qui à f fait correspondre la fonction $v \rightarrow \sum_{u \in F_2^m \,|\, u \preceq v} f(u)$ est involutive.
Or, si on l'applique deux fois à f, on obtient la fonction:

$$v \rightarrow \sum_{w \preceq u} \sum_{u \preceq v} f(w) = \sum_{w \preceq v} f(w) \, |\{u \in F_2^m |\, w \preceq u \preceq v\}|$$

qui est bien la fonction f puisque l'ensemble $\{u \in F_2^m |\, w \preceq u \preceq v\}$ est de cardinal impair si et seulement si $w = v$. ◇

5.2.3 Critères cryptographiques sur les fonctions booléennes

Il y avait jusqu'à maintenant quatre critères importants, et un cinquième critère est apparu très récemment.

Equilibre

Une fonction booléenne utilisée comme fonction de combinaison doit être *équilibrée* (i.e. prendre le même nombre de fois les valeurs 0 et 1), pour éviter que la connaissance du chiffré donne une information statistique sur le clair (en effet, si la fonction "sort" plus de 0 que de 1, on sait que, dans leur majorité, les bits du chiffré sont égaux aux bits correspondants du clair). Cela limite le degré de la fonction à $m - 1$.

Degré algébrique

Puisque la suite générée par le système est périodique, elle est attaquable par l'algorithme de Berlekamp-Massey. Il faut donc qu'elle soit de complexité linéaire élevée. On démontre que la suite générée par une fonction $f(x_1, \cdots, x_m) = \sum_{u \in F_2^m} a_u \left(\prod_{i=1}^{m} {x_i}^{u_i}\right) = \sum_{u \in F_2^m} a_u \, x^u$ prenant en entrées des suites ML générées par des LFSR de longueurs $L_1, \cdots, L_m$ est de complexité linéaire $f(L_1, \cdots, L_m)$, c'est à dire $\sum_{u \in F_2^m} a_u \left(\prod_{i=1}^{m} {L_i}^{u_i}\right)$, où le calcul de la somme est effectué dans $\mathbf{R}$ et non modulo 2. On voit qu'il faut utiliser une fonction de *haut degré algébrique* pour atteindre de hautes complexités linéaires.

Non-corrélation et résilience

Le modèle avec fonction de combinaison peut être attaqué par l'attaque de Siegenthaler ou par l'une de ses améliorations qui ont suivi. Rappelons le principe de l'attaque par corrélation de Siegenthaler: supposons qu'il y ait une corrélation entre la suite pseudo-aléatoire $(s_i)_{i \geq 0}$ générée par le générateur et la sortie $(x_j^i)_{i \geq 0}$ de l'un des LFSR (le $j$ème); la probabilité p_j que $s_i = x_j^i$ est alors différente de $1/2$. En d'autres termes, la probabilité que $s_i = 0$ sachant que $x_j^i = 0$ est différente de $1/2$. De façon équivalente, la probabilité que $f(x) = 0$ sachant que $x_j = 0$ est différente de $1/2$. On peut réaliser une attaque exhaustive sur ce LFSR: on essaie toutes les initialisations possibles jusqu'à ce qu'on observe la corrélation attendue; si l'initialisation est incorrecte, alors la corrélation entre la suite de sortie du LFSR et la suite s_i est celle qu'on observe entre deux suites indépendantes: la probabilité que $s_i = x_j^i$ est égale à $1/2$; sinon, elle est égale à p_j dans le cas d'une attaque à clair connu (dans laquelle on suppose connue une sous-suite du chiffré et la sous-suite correspondante du clair) et elle vaut une probabilité calculable et différente de $1/2$, dans le cas d'une attaque à chiffré seul, en supposant que la probabilité d'apparition du 0 dans le clair est elle-même différente de $1/2$. Une fois l'initialisation du $j$ème LFSR obtenue, on attaque le reste du schéma par une attaque exhaustive, ce qui remplace la complexité de l'attaque exhaustive globale du système, égale à $O(2^{\sum_{i=1}^{m} L_i})$, par $O(2^{L_j} + 2^{\sum_{1 \leq i \leq m / \, i \neq j} L_i})$ (d'où un gain important). Une fonction permet une résistance optimale à cette attaque par corrélation si, pour tout j, la fonction $f(x)$ étant équilibrée, elle reste équilibrée si l'on fixe la valeur de x_j. Nous verrons que c'est équivalent

au fait que l'on a $\sum_{x \in F_2^m} (-1)^{f(x)+x_j} = 0$ pour tout j.
Le même principe d'attaque peut se faire en testant plusieurs LFSR (disons au plus k) au lieu d'un seul. Une fonction permet une résistance optimale à l'attaque consistant à exploiter une corrélation entre la sortie de f et au plus k sorties des LFSR si la fonction $f(x)$ reste équilibrée si l'on fixe les valeurs d'au plus k variables $x_{j_1}, \cdots, x_{j_k}$.

Définition 13 *Une fonction est sans corrélation d'ordre k si, quel que soit le sous ensemble I de cardinal k de* $\{1, \ldots, n\}$ *et quel que soit le vecteur* $a \in F_2^I$*, la restriction de f obtenue en fixant les valeurs* x_i*,* $i \in I$ *à* a_i *a la même distribution de valeurs que f elle-même (i.e. la probabilité qu'elle sorte un 0 est la même). La fonction est dite k-résiliente si elle est de plus équilibrée.*

Une caractérisation fait intervenir une transformation qui joue un rôle très important dans l'étude des fonctions booléennes:

La transformée de Walsh est l'application qui à tout vecteur $a \in F_2^m$ fait correspondre $\widehat{\mathrm{f}}(a) = \sum_{x \in F_2^m} (-1)^{f(x)+a \cdot x}$. La transformée de Walsh est un cas particulier d'une transformation générale appelée la transformée de Fourier.

Définition 14 *Soit* φ *une fonction numérique (i.e. à valeurs dans* $\mathbf{Z}$ *ou* $\mathbf{R}$ *ou* $\mathbf{C}$*) sur* F_2^m*. La transformée de Fourier de* φ *est la fonction*

$$\widehat{\varphi}(s) = \sum_{x \in F_2^m} \varphi(x) \, (-1)^{s \cdot x}$$

où "$\cdot$*" désigne le* produit scalaire *usuel sur* F_2^m*, défini par:*

$$s \cdot x = \sum_{i=1}^{m} s_i x_i \, mod 2.$$

Algorithme:
- Ecrire la table de valeurs de la fonction en mettant les mots binaires de longueur m dans l'ordre naturel
- Soit ψ la fonction sur F_2^{m-1} correspondant a la partie superieure de la table et ψ' celle correspondant a la partie inferieure. Remplacer ψ par $\psi + \psi'$ et ψ' par $\psi - \psi'$ (sommes calculees dans $\mathbf{Z}$).
- Recommencer recursivement sur chacune des moities ainsi obtenues.

Lorsque l'algorithme termine (c'est a dire lorsque on en est arrivé à des fonctions sur F_2^1), les nombres obtenus en face de chaque mot sont les valeurs de la transformee de Fourier.
La complexite est en $O(m2^m)$.

Remarque 8 *Il existe une définition légèrement différente, dans laquelle on divise la somme* $\sum_{x\in F_2^m} \varphi(x)\,(-1)^{s\cdot x}$ *par* $2^{\frac{m}{2}}$. *Nous l'appellerons (pour des raisons qui apparaitront plus loin) la transformation de Fourier normalisée.*

Rappelons une propriété que nous avons déjà rencontrée plusieurs fois sous des formes différentes:

Proposition 4 *Soit E un sous-espace vectoriel de* F_2^m. *On note* 1_E *son indicatrice. Alors, on a:*

$$\widehat{1_E} = |E|\, 1_{E^\perp}.$$

La preuve est laissée au lecteur.
On utilisera souvent cette propriété avec $E = F_2^m$:

$$\widehat{1} = 2^m\, \delta_0$$

où δ_0 est le symbole de Dirac ($\delta_0(x) = 1$ si $x = 0$ et 0 sinon).
On retrouve le résultat utilisé à l'occasion de l'identité de Mac Williams:

Corollaire 1 *[Formule de sommation de Poisson] Pour tout sous-espace vectoriel E de* F_2^m, *on a:*

$$\sum_{s\in E} \widehat{\varphi}(s) = |E| \sum_{x\in E^\perp} \varphi(x).$$

Preuve:

$$\sum_{s\in E} \widehat{\varphi}(s) = \sum_{x\in F_2^m} \varphi(x)\, \widehat{1_E}(x) = |E| \sum_{x\in E^\perp} \varphi(x).$$

◇

La transformée de Fourier a en outre deux propriétés très utiles qui en font un outil beaucoup utilisé en informatique et en mathématiques discrètes:

Proposition 5 *Pour toute fonction* φ:

$$\widehat{\widehat{\varphi}} = 2^m\, \varphi.$$

La transformation de Fourier normalisée est donc involutive.
Preuve:

$$\widehat{\widehat{\varphi}}(x) = \sum_{s\in F_2^m} \widehat{\varphi}(s)\,(-1)^{x\cdot s} = \sum_{s\in F_2^m} \sum_{y\in F_2^m} \varphi(y)\,(-1)^{x\cdot s + s\cdot y} =$$

$$\sum_{y\in F_2^m}\left(\sum_{s\in F_2^m}(-1)^{(x+y)\cdot s}\right)\varphi(y)=\sum_{y\in F_2^m}\widehat{1}(x+y)\,\varphi(y)=2^m\,\varphi(x).$$

◇

On en déduit que la transformée de Fourier est une permutation de l'ensemble des fonctions sur F_2^m à valeurs dans $\mathbf{R}$ ou dans $\mathbf{C}$ (mais pas dans $\mathbf{Z}$, et on ne sait pas caractériser efficacement les transformées de Fourier des fonctions sur F_2^m à valeurs dans $\mathbf{Z}$).

Exercice 18 *Déduire de la proposition précédente qu'une fonction numérique φ est constante si et seulement si sa transformée de Fourier est nulle en tout mot non nul, et qu'elle est constante sauf en 0 si et seulement si sa transformée de Fourier est elle-même constante sauf en 0.*

Avant d'énoncer la deuxième propriété, rappelons ce qu'est le produit de convolution de deux fonctions numériques sur F_2^m:

$$(\varphi\otimes\psi)(x)=\sum_{y\in F_2^m}\varphi(y)\psi(x+y).$$

Cette définition peut se donner plus généralement dans le cadre des fonctions numériques définies sur un groupe abélien (elle s'écrit avec $x-y$ au lieu de $x+y$, mais ici, nous sommes en caractéristique 2).

Proposition 6 *Pour toute fonction φ et toute fonction ψ, on a:*

$$\widehat{\varphi\otimes\psi}=\widehat{\varphi}\,\widehat{\psi}$$

et

$$\widehat{\varphi}\otimes\widehat{\psi}=2^m\,\widehat{\varphi\,\psi}.$$

Preuve:

$$\widehat{\varphi\otimes\psi}(s)=\sum_{x\in F_2^m}(\varphi\otimes\psi)(x)\,(-1)^{s\cdot x}=\sum_{x\in F_2^m}\sum_{y\in F_2^m}\varphi(y)\psi(x+y)\,(-1)^{s\cdot x}=$$

$$\sum_{x\in F_2^m}\sum_{y\in F_2^m}\varphi(y)\psi(x+y)\,(-1)^{s\cdot y+s\cdot(x+y)}=\sum_{y\in F_2^m}\varphi(y)(-1)^{s\cdot y}\left(\sum_{x\in F_2^m}\psi(x+y)\,(-1)^{s\cdot(x+y)}\right)=$$

$$\left(\sum_{y\in F_2^m}\varphi(y)(-1)^{s\cdot y}\right)\left(\sum_{x\in F_2^m}\psi(x)\,(-1)^{s\cdot x}\right)=\widehat{\varphi}(s)\,\widehat{\psi}(s).$$

La première égalité est donc vérifiée. La deuxième s'obtient en appliquant la première aux fonctions $\widehat{\varphi}$ et $\widehat{\psi}$ et en utilisant la proposition 5. ◇

La deuxième égalité appliquée en le mot 0 et pour $\psi = \varphi$ donne la relation dite de *Parseval*:

Corollaire 2 *Pour toute fonction numérique φ:*

$$\sum_{s \in F_2^m} (\widehat{\varphi}(s))^2 = 2^m \sum_{s \in F_2^m} (\varphi(s))^2.$$

Si de plus, φ est à valeurs ± 1, alors:

$$\sum_{s \in F_2^m} (\widehat{\varphi}(s))^2 = 2^{2m}.$$

Propriétés des fonctions sans corrélation et résilientes

Proposition 7 *Une fonction est sans corrélation d'ordre k si et seulement si on a $\widehat{\mathrm{f}}(s) = 0$ pour tout vecteur non nul de poids compris entre 1 et k. Elle est k-résiliente si et seulement si on a $\widehat{\mathrm{f}}(s) = 0$ pour tout veecteur de poids au plus k.*

C'est une conséquence directe de la formule de sommation de Poisson appliquée à la fonction $f(x+u)$ où le vecteur $u \in F_2^m$ coïncide avec a pour les indices appartenant à I (voir la définition 13), et à l'espace vectoriel $E = \{s \in F_2^m / \, s_i = 0, \forall i \notin I\}$. On notera que la fonction $f(x+u)$ a le même ordre de non corrélation que f, car, pour tout vecteur a, on a $\sum_{x \in F_2^m} (-1)^{f(x+u)+a \cdot x} = (-1)^{a \cdot u} \sum_{x \in F_2^m} (-1)^{f(x)+a \cdot x}$. ◇

Exercice 19 *Soit f une fonction k-résiliente qui n'est pas $(k+1)$-résiliente. On suppose sans perte de généralité (quitte à permuter les coordonnées) que $\widehat{\mathrm{f}}(1,\cdots,1,0,\cdots,0) \neq 0$ où $(1,\cdots,1,0,\cdots,0)$ est de poids $k+1$. Montrer que l'une des fonctions g ne dépendant que des variables $x_1,\cdots,x_{k+1}$ qui sont les plus proches de f est alors la fonction $x_1 + \cdots + x_{k+1}$ ou la fonction $x_1 + \cdots + x_{k+1} + 1$. On pourra exprimer la valeur de $g(a_1,\cdots,a_{k+1})$ en fonction des poids des restrictions de f obtenues en fixant les $k+1$ coordonnées de x et en déduire que pour au moins l'une des fonctions g, pour tout $j \leq k+1$, la valeur de g change nécessairement quand on change la valeur de n'importe lequel de ses bits d'entrée.*

On a la borne suivante sur le degré de f:

Proposition 8 *Si f est sans corrélation d'ordre k, alors elle est de degré au plus $m-k$.*
Si f est résiliente d'ordre k, alors:

- *si* $k \leq m-2$*, son degré est au plus* $m-k-1$*,*
- *sinon* $k = m-1$ *et son degré est* 1*.*

Preuve: La preuve est une conséquence directe des propositions 3 et 7. $\diamond$

Remarque 9 *Si* f *est sans corrélation d'ordre* k*, alors son degré est majoré par* $m-k-1$ *sous la condition plus faible que le poids de* f *soit divisible par* 2^{k+1}*.*

La notion de fonction sans corrélation est liée à celle de tableau orthogonal et il existe plusieurs constructions, directes ou récursives, de fonctions sans corrélations.
Citons une construction directe[2], dite de *Maiorana-McFarland*:

Proposition 9 *Soit* r *un entier compris entre* 1 *et* $m-1$*,* g *une fonction booléenne sur* F_2^{m-r} *et* ϕ *une application de* F_2^{m-r} *dans* F_2^r*. Soit* f *la fonction définie sur* F_2^m *par:*

$$\forall x \in F_2^r, \forall y \in F_2^{m-r}, f(x \,|\, y) = x \cdot \phi(y) + g(y)$$

(où "$\cdot$*" désigne ici le produit scalaire dans* F_2^r *et* $|$ *la concaténation).*
Alors f *est résiliente d'ordre* $k \geq inf\{\omega(\phi(y)) | \, y \in F_2^{m-r}\} - 1$*, où* $\omega(\phi(y))$ *désigne le poids de Hamming du mot* $\phi(y)$*.*

Preuve: On a, pour tout mot s de F_2^r et tout mot t de F_2^{m-r}:

$$\widehat{\mathrm{f}}(s \,|\, t) = \sum_{x \in F_2^r} \sum_{y \in F_2^{m-r}} (-1)^{x \cdot \phi(y) + g(y) + s \cdot x + t \cdot y} = \sum_{y \in F_2^{m-r}} (-1)^{g(y) + t \cdot y} \left(\sum_{x \in F_2^r} (-1)^{x \cdot (\phi(y) + s)} \right)$$

(on note de la même façon les produits scalaires dans F_2^r et dans F_2^{m-r}).
D'après la proposition 4, la somme $\sum_{x \in F_2^r} (-1)^{x \cdot (\phi(y)+s)}$ est nulle si et seulement si $s \neq \phi(y)$.
Si s est de poids strictement inférieur au plus petit poids des éléments $\phi(y), y \in F_2^{m-r}$, alors $\widehat{\mathrm{f}}(s \,|\, t)$ est donc nul.
f est donc résiliente d'ordre $k \geq inf\{\omega(\phi(y)) | \, y \in F_2^{m-r}\} - 1$. $\diamond$

2. P. Camion, C. Carlet, P. Charpin et N. Sendrier, *On correlation-immune functions*, Advances in Cryptology, CRYPTO'91, Lecture Notes in Computer Sciences 576, Springer Verlag, pp 86-100 (1992)

Cette proposition permet d'obtenir des fonctions dont les degrés atteignent la borne rappelée ci-dessus:

Exercice 20 *Pour tout r compris entre 1 et $m-2$, montrer qu'il existe des fonctions définies par la proposition précédente qui sont de degré $m-r$ et résilientes d'ordre $r-1$.*

Nonlinéarité

A cause de la formule de Parseval, il n'existe pas de fonction de m variables qui soit m-résiliente. Il y a donc forcément une corrélation entre la sortie de la fonction et un sous-ensemble strict des coordonnées de x. Il est alors préférable que les corrélations non nulles soient aussi faibles que possible, c'est à dire que $\max_{a\in F_2^m} |\widehat{\mathrm{f}}(a)|$ soit aussi petit que possible. On rappelle que la distance de Hamming de f à l'ensemble des fonction affines (i.e. le code de Reed-Muller d'ordre 1) est égale à $2^{m-1} - \frac{1}{2}\max_{a\in F_2^m} |\widehat{\mathrm{f}}(a)|$. Le critère évoqué ci-dessus[3] est donc que cette distance soit aussi grande que possible.

Définition 15 *On appelle nonlinéarité d'une fonction booléenne f et on note N_f sa distance à l'ensemble des fonctions affines.*

On a donc

$$N_f = 2^{m-1} - \frac{1}{2}\max_{a\in F_2^m} |\widehat{\mathrm{f}}(a)|. \tag{5.3}$$

D'après la formule de Parseval, on a $N_f \leq 2^{m-1} - 2^{m/2-1}$ (borne dite universelle). Cette borne est valable pour toutes les fonctions, et elle est atteinte, si m est pair, par les fonctions dites *courbes* (qui sont telles que $\widehat{\mathrm{f}}(a) = \pm 2^{m/2}$ pour tout a). En particulier (prendre $a=0$) elles ne sont pas équilibrées. Par contre, leurs dérivées sont équilibrées:

Exercice 21 *Montrer qu'une fonction booléenne est courbe si et seulement si toutes ses dérivées $D_a f(x) = f(x) + f(x+a)$, $a \neq 0$ sont équilibrées.*

Un exemple de fonction courbe: $f(x,y) = x\cdot\pi(y)) + h(y)$, où $x,y \in F_2^{n/2}$, où π est une permutation quelconque de $F_2^{n/2}$ et où h est une fonction booléenne quelconque (f ainsi définie est une fonction dite de Maiorana-McFarland).

3. De plus, toute approximation affine fournit un distingueur et don à une faiblesse du système

Dans le cas des fonctions résilientes, on peut prouver une meilleure borne que la borne universelle.

Proposition 10 (Borne de Sarkar-Maitra) *Soit f une fonction k-résiliente ($k \leq m-2$).*
1. Alors, pour tout $a \in F_2^m$, le coefficient de Walsh $\widehat{\mathrm{f}}(a)$ est divisible par 2^{k+2}.
2. On en déduit que $N_f \leq 2^{m-1} - 2^{k+1}$. De plus, si $k \leq m/2-2$ avec m pair, alors on a $N_f \leq 2^{m-1} - 2^{m/2-1} - 2^{k+1}$.

Preuve: 1. On démontre la divisibilité de $\widehat{\mathrm{f}}(a)$ par 2^{k+2} par récurrence sur le poids de Hamming de $a \in F_2^m$. Pour $w_H(a) \leq k$, elle est évidente. Pour $w_H(a) = k+1$, puis pour $w_H(a) > k+1$, on la déduit de la formule de sommation de Poisson avec $E = \{x \in F_2^m / \, x \preceq a\}$.
2. On en déduit que, d'après la relation (5.3), N_f est divisible par 2^{k+1}. Comme on sait que $N_f < 2^{m-1}$, on en déduit $N_f \leq 2^{m-1} - 2^{k+1}$. De plus, comme on sait que toute fonction courbe est non-équilibrée, on a $N_f < 2^{m-1} - 2^{m/2-1}$. Donc N_f est inférieur ou égal au plus grand multiple de 2^{k+1} qui est strictement inférieur à $2^{m-1} - 2^{m/2-1}$. Si $k \leq m/2-2$, alors on a $N_f \leq 2^{m-1} - 2^{m/2-1} - 2^{k+1}$, si m est pair. $\diamond$

Remarque: d'après la relation de Parseval et la relation (5.3), N_f est majoré par $2^{m-1} - \frac{2^{m-1}}{\sqrt{\sum_{i=k+1}^{m} \binom{m}{i}}}$. Donc N_f est majoré par le plus grand entier divisible par 2^{k+1} qui est inférieur ou égal à $2^{m-1} - \frac{2^{m-1}}{\sqrt{\sum_{i=k+1}^{m} \binom{m}{i}}}$.

Un critère lié aux attaques algébriques

Si une fonction de filtrage (par exemple) est telle qu'il existe deux fonctions g et h de bas degrés (disons de degrés au plus d) telles que $f * g = h$ (où $*$ désigne le produit) et $g \neq 0$, alors on en déduit l'attaque suivante: notons n la longueur du LFSR filtré par f; on considère f comme une fonction de n variables, même si elle dépend en fait de moins de variables; on sait qu'il existe une application linéaire L de F_2^n dans F_2^n telle que la sortie du générateur soit $s_i = f(L^i(u_1, \ldots, u_n))$, où $u_1, \ldots, u_n$ est l'initialisation du LFSR (ceci est vrai plus généralement pour tous les automates linéaires combinés par une fonction booléenne). On a alors, pour tout i: $s_i * g(L^i(u_1, \ldots, u_n)) = h(L^i(u_1, \ldots, u_n))$. Cette équation en $u_1, \ldots, u_n$ est de

degré au plus d, puisque L est linéaire. Une fonction offre une résistance optimale à cette attaque s'il n'existe pas g et h de bas degrés, $g \neq 0$, telles que $f * g = h$.

Exercice 22 *Montrer qu'il existe $g \neq 0$ et h telles que $f * g = h$ si et seulement si il existe $g \neq 0$, telle que $f * g = 0$ ou $(f + 1) * g = 0$.*

On appelle immunité algébrique de f et on note $AI(f)$ le degré minimal des fonctions g non nulles telles que $f * g = 0$ ou $(f + 1) * g = 0$.

Exercice 23 *Montrer que $AI(f) \leq \lceil n/2 \rceil$.*

Chapitre 6

Les schémas de chiffrement par blocs

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

Dans un système par blocs, chaque clair est découpé en blocs de même longueur et chiffré bloc par bloc. La longueur l des clés doit être suffisante pour que l'attaque exhaustive (attaque à chiffré seul si on a un moyen de différencier les clairs des messages aléatoires, et attaque à clair connu sinon) consistant à déchiffrer le chiffré avec toutes les clés possibles jusqu'à l'obtention du clair, soit irréaliste ($l \geq 80$). La longueur n des blocs doit également être suffisante pour éviter les attaques dites par dictionnaire consistant à s'aider d'un pré-calcul (partiel) des chiffrés des 2^n blocs possibles.

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

Les exemples historiques de chiffrement (par transposition et par substitution) vus en début de polycopié sont des chiffrements par blocs. La

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

Définition 16 *On appelle chiffrement produit un chiffrement par blocs qui combine plusieurs transformations élémentaires (substitutions, transpositions, opérations linéaires ou arithmétiques).*
Un chiffrement itératif résulte de l'application itérée d'un chiffrement (en général un chiffrement produit). Chaque itération est appelée un tour (round en anglais). Chaque tour fait intervenir une sous-clé qui est dérivée (on dit souvent cadencée) de la clé principale.

Les schémas de Feistel et le DES

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

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

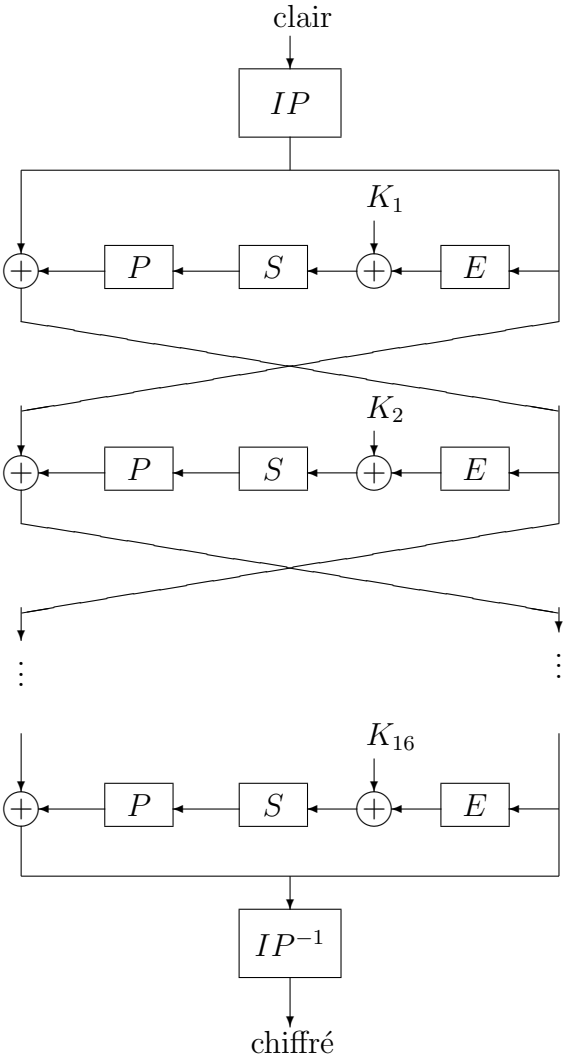
où f est un chiffrement produit.
Tout chiffrement de Feistel est inversible, que f soit une bijection ou non.
Le permuté (R,L) du clair s'obtient à partir de celui (R',L') du chiffré en lui appliquant un chiffrement de Feistel de même schéma et en prenant comme liste de sous-clés de déchiffrement la même que celle des sous-clés de chiffrement, mais dans l'ordre inverse.

Le DES qui a été le standard (proposé par le National Bureau of Standards américain et développé par IBM) utilisé de 1977 à 2000 dans le monde entier, est un schéma de Feistel. L'algorithme de chiffrement est complètement spécifié mais fait intervenir des *boites S* dont les critères de conception n'ont pas été rendus publics. Certains pensent que la NSA (National Security

Agency) qui avait participé à l'opération avait introduit des *trappes* lui permettant de décrypter plus facilement les messages, mais cela n'a jamais pu être prouvé puisqu'aucune cryptanalyse sensiblement plus efficace que l'attaque exhaustive n'a pu être trouvée[1]. En janvier 98, une attaque exhaustive a été réalisée en 39 jours sur 10 000 pentiums en parallèle. Et en juillet 98, un autre attaque a pris 56 heures avec une machine dédiée de 250 000 dollars. L'adoption de l'AES s'imposait donc.

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

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



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

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

La cryptanalyse différentielle du DES date de 1990 et elle est due à Biham et Shamir. C'est une attaque à clair choisi, efficace jusqu'à 8 tours. Pour les 16 tours du DES, elle nécessite 2^{47} couples (clair,chiffré) choisis et on lui préfère en fait l'attaque exhaustive. Le principe est de chercher à déterminer la clé du dernier tour, puis celle-ci étant connue, d'attaquer de la même façon l'avant dernier tour et ainsi de suite.
Nous montrons donc comment attaquer le dernier tour.
Pour chaque clair m, on note $Y(0)$ ce clair et $Y(1),\cdots,Y(r)$ les chiffrés produits par les tours successifs (pour le DES on a $r = 16$) avec les clés $K_1,\cdots,K_r$ déduites de la clé K. On étudie d'abord en probabilité le comportement de la différence $\Delta Y(r-1) = Y(r-1) \oplus Y'(r-1)$ correspondant à deux clairs $Y(0) = m$ et $Y'(0) = m'$ dont la différence $\Delta Y(0) = m \oplus m'$ est constante (m variant aléatoirement dans l'espace des clairs et K variant aléatoirement dans l'espace des clés). Pour chaque couple $(\alpha,\beta) \in F_2^{64} \times F_2^{64}$, la probabilité conditionnelle $P_{\alpha,\beta} = P(\Delta Y(r-1) = \beta \,|\, \Delta Y(0) = \alpha)$ peut être calculée mathématiquement (sans avoir à réaliser les chiffrements relatifs aux différentes clés, ce qui serait impossible). On choisit alors un couple (α,β) tel que cette probabilité soit maximale. L'attaque est d'autant plus efficace que ce maximum est plus éloigné de la probabilité moyenne, égale à 2^{-64}.
L'algorithme de l'attaque consiste à itérer le processus suivant:

1. Choisir un message m et soumettre m et $m \oplus \alpha$ au chiffrement. On obtient c et c' comme chiffrés;
2. Déterminer toutes les valeurs possibles de la clé du r-ième tour telles que $\Delta Y(r-1) = \beta$ et $\Delta Y(r) = c \oplus c'$ (nous montrerons plus loin comment cela est possible);
3. Incrémenter un compteur pour chacune des valeurs possibles de K_r trouvées à l'étape précédente;

jusqu'à ce qu'une valeur de clé ait été incrémentée significativement plus que les autres.

Montrons que cela se produit nécessairement pour la bonne clé: dans tous les cas où l'hypothèse $\Delta Y(r-1) = \beta$ n'est pas vérifiée[2], la liste des clés produite par l'étape 2 est aléatoire. Mais quand cette hypothèse est vérifiée, la bonne clé K_r est systématiquement incrémentée. Or on a supposé que la probabilité que cette hypothèse soit vérifiée est plus grande que la probabilité uniforme. Donc la bonne clé est incrémentée plus souvent qu'une clé quelconque (exercice!). Ce principe est général à tous les chiffrements par blocs.
Montrons maintenant comment l'étape 2 est réalisée pour le DES.
Rappelons qu'on se place dans le cas où $\Delta Y(r-1) = \beta$. Notons ΔE la différence entre les sorties de la boite E du dernier tour correspondant respectivement aux clairs m et $m+\alpha$. On a $\Delta E = E(Y_L(r-1)) \oplus E(Y'_L(r-1)) = E(Y_L(r-1) \oplus Y'_L(r-1)) = E(\beta_L)$ (E étant linéaire). Notons ΔS la différence à la sortie de la boite S. On a $\Delta S = P^{-1}(Y_R(r-1) \oplus Y_L(r)) \oplus P^{-1}(Y'_R(r-1) \oplus Y'_L(r))$, soit: $\Delta S = P^{-1}(Y_R(r-1) \oplus Y'_R(r-1) \oplus Y_L(r) \oplus Y'_L(r)) = P^{-1}(\beta_R \oplus Y_L(r) \oplus Y'_L(r))$ (P^{-1} étant linéaire). Notons $x_1, \cdots, x_l$ les solutions de l'équation:

$$S(x) \oplus S(x \oplus \Delta E) = \Delta S.$$

Ce sont les vecteurs susceptibles d'être les entrées de la boite S correspondant (par exemple) au clair m. Les clés à incrémenter sont donc les valeurs de K_r telles que $x_i = E(Y_R(r)) \oplus K_r$.

La cryptanalyse linéaire date de 1993 et elle est due à Matsui. C'est une attaque à clair connu. Nous présentons deux versions de l'attaque: l'attaque sur le chiffrement global et l'attaque sur le dernier tour.
Dans l'attaque sur le chiffrement global, on cherche d'abord mathématiquement les triplets $(\alpha,\beta,\gamma) \in F_2^{64} \times F_2^{64} \times F_2^{56}$ tels que, en faisant parcourir aléatoirement par X l'ensemble des clairs, par K l'ensemble des clés et en notant Y le chiffré de X avec la clé K, la probabilité $p_{\alpha,\beta,\gamma}$ que le bit égal à $\alpha \cdot X \oplus \beta \cdot Y \oplus \gamma \cdot K$ (où "$\cdot$" désigne le produit scalaire) soit nul soit éloignée de $1/2$, c'est à dire telle que $\alpha \cdot X \oplus \beta \cdot Y \oplus \gamma \cdot K$ soit égal à $\epsilon \in F_2$ avec une probabilité nettement supérieure à $1/2$. L'algorithme de l'attaque consiste à itérer le processus suivant:

1. Choisir un tel triplet (α,β,γ);
2. Pour N couples (clair,chiffré), correspondant évidemment à la clé qu'on cherche à déterminer, calculer le nombre T des couples tels que $\alpha \cdot X \oplus$

2. On peut en fait éliminer tous les cas où la partie droite ΔY_R de ΔY n'est pas égale à la partie droite β_R de β, puisqu'on connait $\Delta Y_R = c_L \oplus c'_L$.

$\beta \cdot Y = 0$. Si $T > N/2$ alors $\gamma \cdot K = \epsilon$ et sinon $\gamma \cdot K = \epsilon \oplus 1$.
Matsui a calculé le taux de succès de cette attaque selon les valeurs de N. Par exemple, si $N = \frac{1}{(p-1/2)^2}$ alors le taux de succès est de 97,7%.

Cette attaque ne donne qu'une information partielle sur la clé: à l'issue de l'attaque, on sait que la somme des bits de la clé qui correspondent aux coordonnées non nulles de γ est égale à une constante η (où η est égal à ϵ ou à $\epsilon \oplus 1$).

Une méthode plus efficace, si on ne dispose pas d'assez de triplets (α,β,γ), consiste à attaquer le dernier tour comme dans le cas de l'attaque différentielle: on cherche d'abord mathématiquement les triplets $(\alpha,\beta,\gamma) \in F_2^{64} \times F_2^{64} \times F_2^{56}$ tels que, en faisant parcourir aléatoirement par X l'ensemble des clairs, par K l'ensemble des clés et en notant $Y(r-1)$ le chiffré de X à l'issue de l'avant-dernier tour avec la clé K, la propabilité $p_{\alpha,\beta,\gamma}$ que le bit égal à $\alpha \cdot X \oplus \beta \cdot Y(r-1) \oplus \gamma \cdot K$ (où "$\cdot$" désigne le produit scalaire) soit nul soit éloignée de 1/2, c'est à dire telle que

$$\alpha \cdot X \oplus \beta \cdot Y(r-1) \oplus \gamma \cdot K = \epsilon \tag{6.1}$$

avec une probabilité nettement supérieure à 1/2. Notons K_r la clé (inconnue) du dernier tour, la relation (6.1) s'écrit:

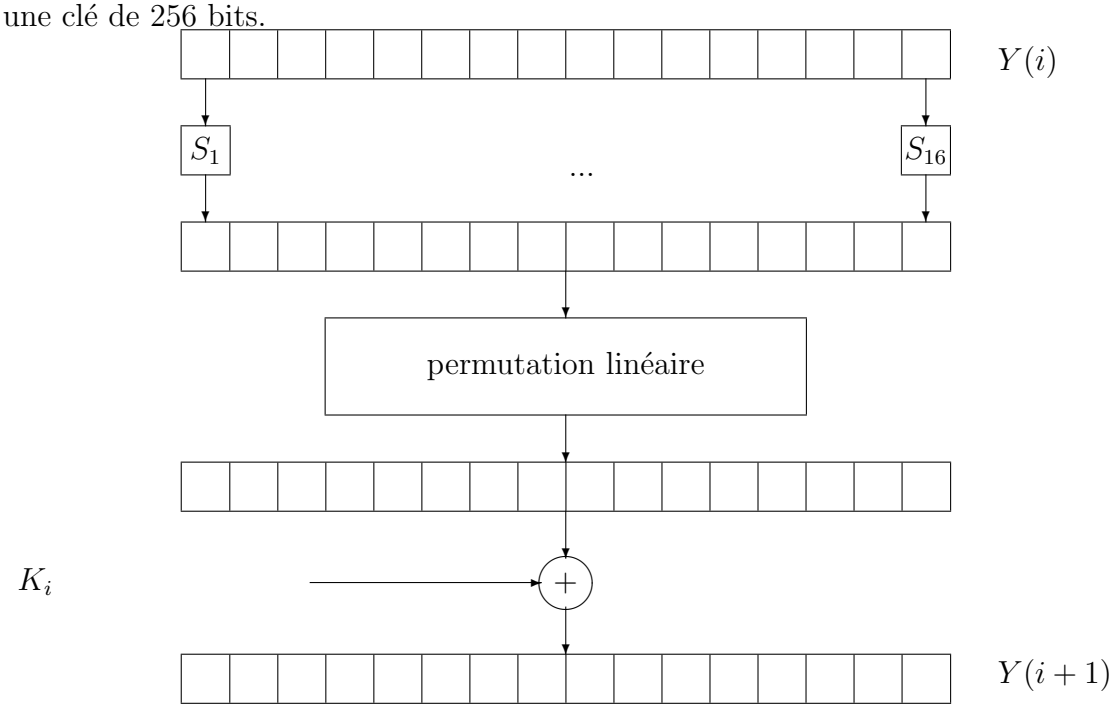
$$\alpha \cdot X \oplus \beta \cdot F^{-1}(Y(r),K_r) \oplus \gamma \cdot K = \epsilon.$$

On va donc calculer $\alpha \cdot X \oplus \beta \cdot F^{-1}(Y(r),k_r)$ pour N couples (clair,chiffré) et pour toutes les valeurs possibles k_r de la clé du dernier tour. Si la valeur essayée est exacte (i.e. $k_r = K_r$), alors le nombre de couples (X,$Y(r)$) tels que $\alpha \cdot X \oplus \beta \cdot F^{-1}(Y(r),k_r) = 1$ est soit anormalement faible (si $\gamma \cdot K = \epsilon$) que soit anormalement fort (si $\gamma \cdot K = \epsilon \oplus 1$). On en déduit donc K_r et on passe à l'attaque du tour précédent.

Les schémas de permutation-substitution et l'AES

L'Advanced Encryption Standard a fait l'obet d'un appel d'offre datant de 1997. Il s'agissait de remplacer le DES dont la taille des clés était devenue trop petite pour les performances des ordinateurs modernes. Les spécifications étaient une longueur de blocs de 128 bits (ou de 256 bits) et une longueur de clé paramétrable: 128 ou 192 ou 256 bits. Parmi les 15 candidats, le candidat retenu (en 2000) se nomme RIJNDAEL (mais on l'appelle simplement l'AES). Il est dû à deux chercheurs Belges, Rijmen et Daemen. C'est un

chiffrement itératif, mais contrairement à 9 autres candidats, ce n'est pas un chiffrement de Feistel. C'est un chiffrement par substitution-permutation: à chaque tour, le chiffré $Y(i)$ produit par le tour précédent subit une substitution non-linéaire qui assure la confusion puis une permutation linéaire qui assure la diffusion, puis la clé du tour est ajoutée bit à bit pour donner $Y(i+1)$. Le nombre de tours est 10 pour une clé de 128 bits et de 14 pour une clé de 256 bits.



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

coefficients dans le corps F_{256}:

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

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

La fonction de substitution (la boite S): chaque octet est considéré comme un élément du corps F_{256}. La boite S est constituée de 16 boites identiques consécutives agissant chacune sur un octet. Chaque boite S_i consiste en l'application $F : x \in F_{256} \mapsto x^{254} \in F_{256}$. Notons que si $x = 0$ alors $x^{254} = 0$ et si $x \neq 0$ alors $x^{254} = \frac{1}{x}$. Le résultat de cette transformation subit ensuite une application affine. Cette boite S permet au système de résister à l'attaque différentielle et à l'attaque linéaire. La raison de ce choix est que la fonction $F : x \in F_{256} \mapsto x^{254} \in F_{256}$ est telle que l'équation $F(x) + F(x + a) = b$ admet au plus 4 solutions (cela évite l'existence de différentielles de probabilités élevées) et que l'équation $a \cdot X \oplus b \cdot F(X) = 0$ admet un nombre de solutions proche de 2^7 pour tout $a \neq 0$ et tout $b \neq 0$ (cela rend couteuse l'attaque linéaire).

6.1 Fonctions booléennes vectorielles pour les schémas par blocs

Soient n et m deux entiers positifs. Les applications de F_2^n dans F_2^m, sont appelées des (n,m)-fonctions. Une (n,m)-fonction F étant donnée, les fonctions Booléennes $f_1, \ldots, f_m$ définies, pour tout $x \in F_2^n$, par

$$F(x) = (f_1(x), \ldots, f_m(x))$$

sont appelées les fonctions coordonnées de F. Quand les nombres n et m ne sont pas spécifiés, on appelle les (n,m)-fonctions des fonctions Booléennes vectorielles ou des boites-S.
D'après ce qui a été rappelé à la section précédente, une (n,m)-fonction F contribue à une résistance optimale à l'attaque différentielle si, pour tout α

non nul dans F_2^n et tout β dans F_2^m, l'équation $F(x) + F(x+\alpha) = \beta$ admet aussi peu de solutions que possible. Le minimum est clairement 2.

Définition 17 *Une (n,n)-fonction F est dite presque parfaitement nonlinéaire (notation: APN, pour "almost perfect nonlinear") si, pour tout α non nul dans F_2^n et tout β dans F_2^m, l'équation $F(x) + F(x+\alpha) = \beta$ admet au plus 2 solutions (c'est à dire, soit deux solutions soit aucune).*

Une (n,n)-fonction F contribue à une résistance optimale à l'attaque linéaire si, pour tout α dans F_2^n et tout β non nul dans F_2^m, la fonction booléenne $\alpha \cdot x + \beta \cdot F(x)$ est de poids aussi proche de 2^{n-1} que possible.

Définition 18 *La nonlinéarité d'une (n,m)-fonction F est égale au poids minimal des fonctions $\beta \cdot F(x) + \alpha \cdot x + \epsilon$, où $\beta \in F_2^{m*}$, $\alpha \in F_2^n$ et $\epsilon \in F_2$.*

La nonlinéarité $\mathcal{NL}(F)$ d'une (n,m)-fonction F est donc égale à la nonlinéarité minimale des fonctions $\beta \cdot F$, où $\beta \neq 0$. D'après ce qui a été vu au chapitre précédent, on a donc:

$$\mathcal{NL}(F) = 2^{n-1} - \frac{1}{2} \max_{\beta \in \mathcal{B}^{m*};\ \alpha \in \mathcal{B}^n} \left| \sum_{x \in \mathcal{B}^n} (-1)^{\beta \cdot F(x) \oplus \alpha \cdot x} \right|$$

et la nonlinéarité est bornée supérieurement par $2^{n-1} - 2^{n/2-1}$. Une (n,m)-fonction est dite courbe si elle atteint cette borne, c'est à dire si toutes les fonctions booléennes $\beta \cdot F$, où $\beta \neq 0$, sont courbes.

Un exemple de $(n,n/2)$-fonction courbe: $F(x,y) = L(x\,\pi(y)) + H(y)$, où le produit $x\,\pi(y)$ est calculé dans le corps $F_{2^{n/2}}$, où L est une application linéaire ou affine surjective (et donc équilibrée) de $F_{2^{n/2}}$ dans F_2^m, où π est une permutation quelconque de $F_{2^{n/2}}$ et où H est une $(\frac{n}{2},m)$-fonction quelconque (F ainsi définie est une fonction dite de Maiorana-McFarland).

D'après le résultat prouvé à l'exercice 1 du TD 2, une fonction courbe ne peut donc pas être équilibrée (i.e. avoir une sortie uniformément distribuée), ce qui la rend impropre à une utilisation cryptographique directe. D'après l'exercice 21 et le résultat prouvé à l'exercice 1 du TD 2, F est courbe si et seulement si toutes ses dérivées $D_a F(x) = F(x) + F(x+a)$ sont équilibrées. On dit que F est alors également parfaitement nonlinéaire: cela traduit le fait que le nombre maximal de solutions de l'équation $F(x) + F(x+a) = b$, $a \in F_2^{n*}$, $b \in F_2^m$, est minimal (égal à 2^{n-m}). Mais on démontre que les fonctions courbes n'existent que pour n pair (ce qui est évident) et $m \leq n/2$.

Exercice 24 *Soit n un entier pair, m un entier positif et F une (n,m)-fonction courbe. Montrer que pour tout $b \in F_2^m$, le cardinal de $F^{-1}(b)$ est*

égal à $2^{-m}\sum_{x\in F_2^n;v\in F_2^m}(-1)^{v\cdot(F(x)+b)}$. *En notant* F_v *la fonction booléenne* $x\mapsto v\cdot F(x)$ *et, pour tout* $v\in F_2^{n*}$, $\widetilde{F_v}$ *sa duale, montrer que le cardinal de* $F^{-1}(b)$ *est alors égal à* $2^{n-m}+2^{\frac{n}{2}-m}\sum_{v\in F_2^{n*}}(-1)^{\widetilde{F_v}(0)\oplus v\cdot b}$. *En déduire que* $m\leq n/2$.
La borne universelle doit donc pouvoir être améliorée pour $m>n/2$. On n'a su l'améliorer que pour $m=n$:

Théorème 3 *Soit* F *une* (n,m)*-fonction quelconque. Alors:*

$$\mathcal{NL}(F)\leq 2^{n-1}-\frac{1}{2}\sqrt{3\times 2^n-2-2\frac{(2^n-1)(2^{n-1}-1)}{2^m-1}}.$$

Preuve. On a:

$$\max_{\beta\in F_2^{m*},\alpha\in F_2^n}\left(\sum_{x\in F_2^n}(-1)^{\beta\cdot F(x)\oplus\alpha\cdot x}\right)^2\geq\frac{\sum_{\beta\in F_2^{m*},\alpha\in F_2^n}\left(\sum_{x\in F_2^n}(-1)^{\beta\cdot F(x)\oplus\alpha\cdot x}\right)^4}{\sum_{\beta\in F_2^{m*},\alpha\in F_2^n}\left(\sum_{x\in F_2^n}(-1)^{\beta\cdot F(x)\oplus\alpha\cdot x}\right)^2}. \tag{6.2}$$

La relation de Parseval implique que pour tout $\beta\in F_2^m$:

$$\sum_{\alpha\in F_2^n}\left(\sum_{x\in F_2^n}(-1)^{\beta\cdot F(x)\oplus\alpha\cdot x}\right)^2=2^{2n}.$$

De plus:

$$\begin{aligned}
&\sum_{\beta\in F_2^m,\alpha\in F_2^n}\left(\sum_{x\in F_2^n}(-1)^{\beta\cdot F(x)\oplus\alpha\cdot x}\right)^4\\
&=\sum_{x,y,z,t\in F_2^n}\left[\sum_{\beta\in F_2^m}(-1)^{\beta\cdot(F(x)+F(y)+F(z)+F(t))}\right]\left[\sum_{\alpha\in F_2^n}(-1)^{\alpha\cdot(x+y+z+t)}\right]\\
&=2^{n+m}\left|\left\{(x,y,z,t)\in F_2^{4n}/\ \left\{\begin{array}{ll}x+y+z+t & =0\\ F(x)+F(y)+F(z)+F(t) & =0\end{array}\right.\right\}\right|\\
&=2^{n+m}|\{(x,y,z)\in F_2^{3n}/\ F(x)+F(y)+F(z)+F(x+y+z)=0\}| \qquad (6.3)\\
&\geq 2^{n+m}|\{(x,y,z)\in F_2^{3n}/\ x=y \text{ ou } x=z \text{ ou } y=z\}|. \qquad (6.4)
\end{aligned}$$

Cela prouve le résultat, car $|\{(x,y,z)/\,x=y \text{ or } x=z \text{ or } y=z\}|=3\cdot|\{(x,x,y)/\,x,y\in F_2^n\}|-2\cdot|\{(x,x,x)/\,x\in F_2^n\}|=3(2^{2n})-2(2^n)$. ◇

Pour $m = n$ (qui est, de nos jours, par exemple pour l'AES, le cas pratique), on a donc $\mathcal{NL}(F) \leq 2^{n-1} - 2^{\frac{n-1}{2}}$.

Définition 19 *Une (n,n)-fonction F est dit presque courbe (notation: AB, pour "almost bent") si* $\mathcal{NL}(F) = 2^{n-1} - 2^{\frac{n-1}{2}}$.

Exercice 25 *Montrer que si une fonction est AB alors elle est APN.*

Exercice 26 *Montrer que les deux notions de APN et AB sont stables par composition à droite et à gauche par un automorphisme affine, par ajout d'un endomorphisme affine, et par passage à l'inverse (dans le cas, bien sûr, où F est bijective).*

Exemples connus de fonctions APN et AB: les seuls exemples connus (aux transformations ci-dessus près) sont des fonctions définies dans le corps F_{2^n} (qu'on peut identifier à F_2^n) et de la forme $F(x) = x^s$.

Fonctions APN:

- $s = 2^n - 2$, pour n impair. $F(x)$ égale $\frac{1}{x}$ si $x \neq 0$ et 0 sinon. L'équation $x^{2^n-2} + (x+1)^{2^n-2} = b$ ($b \neq 0$) admet 0 et 1 pour solutions si et seulement si $b = 1$; et elle admet des solutions différentes de 0 et 1 si et seulement si il existe $x \neq 0{,}1$ tel que $\frac{1}{x} + \frac{1}{x+1} = b$, c'est à dire $x^2 + x = \frac{1}{b}$.

  Exercice 27 *Montrer qu'un tel x existe si et seulement si* $tr\left(\frac{1}{b}\right) = 0$.

  Donc F st APN si et seulement si $tr(1) = 1$, c'est à dire si n est impair.
- $s = 2^h + 1$ avec $pgcd(h,n) = 1$ et $1 \leq h \leq \frac{n-1}{2}$ (exercice!);
- $s = 2^{2h} - 2^h + 1$ avec $pgcd(h,n) = 1$ et $2 \leq h \leq \frac{n-1}{2}$;
- $s = 2^{\frac{4n}{5}} + 2^{\frac{3n}{5}} + 2^{\frac{2n}{5}} + 2^{\frac{n}{5}} - 1$, avec n divisible par 5.
- $s = 2^{(n-1)/2} + 3$ (n impair);
- $s = 2^{(n-1)/2} + 2^{(n-1)/4} - 1$, où $n \equiv 1 \pmod 4$;
- $s = 2^{(n-1)/2} + 2^{(3n-1)/4} - 1$, où $n \equiv 3 \pmod 4$.

Fonctions AB:

- $s = 2^h + 1$ avec $pgcd(h,n) = 1$ et $1 \leq h \leq \frac{n-1}{2}$.
- $s = 2^{2h} - 2^h + 1$ avec $pgcd(h,n) = 1$ et $2 \leq h \leq \frac{n-1}{2}$.
- $s = 2^{(n-1)/2} + 3$.
- $s = 2^{(n-1)/2} + 2^{(n-1)/4} - 1$, où $n \equiv 1 \pmod 4$.
- $s = 2^{(n-1)/2} + 2^{(3n-1)/4} - 1$, where $n \equiv 3 \pmod 4$.

Chapitre 7

Codes Cycliques

7.1 Introduction: une présentation élémentaire des codes BCH 2-correcteurs

On a vu que le code de Hamming de longueur $n = 2^m - 1$ nécessite m bits de contrôle pour corriger une erreur. Notons $W_1, \ldots, W_n$ les n vecteurs binaires non nuls de longeur m. La matrice de contrôle du code de Hamming de longueur $n = 2^m - 1$ est:

$$H_m = [W_1, \ldots, W_n].$$

Le code BCH 2-correcteur de même longueur va nécessiter $2m$ bits de contrôle pour corriger 2 erreurs. Pour trouver un tel code, on cherche donc un code dont la matrice de parité H'_m est une matrice $2m \times n$. Il paraît naturel de prendre pour les m prmières lignes de cette matrice les m lignes de H_m. Le code BCH 2-correcteur est donc inclus dans le code de Hamming; il est au moins 1-correcteur.

Il existe une fonction f de l'ensemble $\{W_1, \ldots, W_n\}$ sur ${F_2}^m$ telle que H'_m puisse s'écrire sous la forme:

$$H'_m = \begin{bmatrix} W_1 & W_2 & \ldots & W_n \\ f(W_1) & f(W_2) & \ldots & f(W_n) \end{bmatrix}.$$

Notons W'_i le i-ème vecteur-colonne de H'_m et traduisons maintenant le fait que le code doit être 2-correcteur. Supposons que 2 erreurs se soient produites, aux indices i et j $(i \neq j)$. Le syndrome, noté en colonne, est:

$$S = W'_i + W'_j.$$

Posons $S = (z_1,z_2)$ où $z-1$ et z_2 sont des vecteurs colonnes de longueur m. Nous savons que le système d'équations:

$$\begin{cases} W_i + W_j & = & z_1 \\ f(W_i) + f(W_j) & = & z_2 \end{cases}$$

admet au moins une solution (i,j), puisque S est le syndrome d'un mot-erreur de poids 2, et il nous faut trouver f de sorte qu'il y ait unicité de cette solution.

Lorsque nous avons ce type d'équation à résoudre dans un ensemble de nombres (**Q**, **R** ou **C**), nous utilisons non seulement les opérations d'addition et de soustraction mais aussi celles de multiplication et de division. C'est ce que nous allons faire ici aussi pour les vecteurs. Il nous faut donc introduire une opération de multiplication des vecteurs pour laquelle tout élément non nul puisse être inversé. En d'autres termes, il nous faut mettre sur $\{0,1\}^m$ une structure de corps.

Pour tout entier positif m, il existe sur $\{0,1\}^m$ une structure de corps définie comme suit:

on choisit un polynôme $P(x)$ de degré m à coefficients dans le corps à deux éléments $GF(2) = \{0,1\}$, qui soit irréductible (i.e. tel qu'il n'existe pas deux polynômes Q et R non constants tels que $P = QR$). Tout élément $a = (a_1, \ldots, a_m)$ de $GF(2)^m$ est identifié au polynôme $F_a(x) = a_1 + a_2 x + \ldots + a_m x^{m-1}$. On a bien $F_{a+b}(x) = F_a(x) + F_b(x)$ (l'addition se faisant bien sûr modulo 2). L'opération de multiplication se fait ainsi:

soient a et b deux mots, on calcule le polynôme $F_a(x)\,F_b(x)$ et on le divise par $P(x)$; on obtient un quotient $Q(x)$ et un reste $R(x)$ tels que $F_a(x)\,F_b(x) = P(x)Q(x) + R(x)$, $d^\circ(R) < d^\circ(P) = m$. Le produit de a et de b est par définition le mot c tel que $F_c(x) = R(x)$. L'opération de multiplication ainsi définie et l'opération d'addition confèrent à $GF(2)^m$ une structure de corps. De plus, ce corps est cyclique: il existe un mot α (dit *primitif*) tel que l'ensemble des mots non nuls soit égal à l'ensemble $\{\alpha^i, i = 1, \ldots, 2^m - 1\}$. Cet élément α (qui n'est pas unique) est d'ordre 2^m (i.e. $\alpha^{2^m} = \alpha$, ou encore $\alpha^{2^m-1} = \mathbf{1}$, où $\mathbf{1}$ est l'élément neutre pour la multiplication, c'est à dire le mot $10\ldots0$).

Le corps ainsi défini s'appelle le corps de Galois d'ordre 2^m et se note $GF(2^m)$ (notation anglo-saxonne) et parfois F_{2^m} (notation française). α s'appelle un élément primitif de ce corps.

Exercice 28 *Calculer les tables d'opérations du corps à* 16 *éléments ainsi construit, en prenant pour polynôme irréductible de degré* 4 *le polynôme:*

$P(x) = 1 + x + x^4$. Montrer qu'on peut prendre pour α le mot 0100 *(en calculer toutes ses puissances). Réécrire alors les tables en notant les mots sous la forme:* 0, $\alpha^i, i = 0,1,\ldots,2^m - 2$.

Remarque 10 *Conception de circuits électroniques pour calculer dans les corps finis: les blocs de base sur les bits sont (voir la figure ci-dessous) le stockage (flip-flop), l'additionneur (XOR) et le multiplieur (AND):*

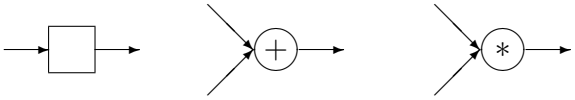


FIG. 7.1 – *blocs de base*

Par exemple, dans $GF(16)$ le circuit pour représenter la multiplication par α sera le suivant:
En effet, si les 4 *flip-flop contiennent initialement a_0,a_1,a_2,a_3 et représentent*

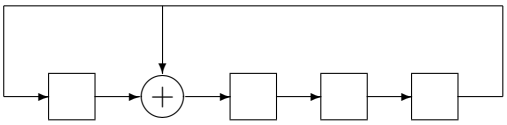


FIG. 7.2 – *multiplication par α*

ainsi l'élément $a_0+a_1\alpha+a_2\alpha^2+a_3\alpha^3$, alors, un instant plus tard, ils contiennent: a_3,a_0+a_3,a_1,a_2 et représentent bien $\alpha(a_0+a_1\alpha+a_2\alpha^2+a_3\alpha^3) = a_0\alpha+a_1\alpha^2+a_2\alpha^3+a_3(1+\alpha) = a_3+(a_0+a_3)\alpha+a_1\alpha^2+a_2\alpha^3$.
Un tel circuit s'appelle un circuit linéaire avec feedback, d'ordre 4*, à une entrée (et invariant dans le temps).*

Voyons maintenant ce qu'on peut choisir comme fonction f. D'après ce qui précède, on peut prendre $W_i = \alpha^i, i = 0,\ldots,2^m - 2$.
• $f(x) = \beta x$ (c'est à dire $f(\alpha^i) = \beta\,\alpha^i$) où β est un élément de $GF(2^m)$ ne peut pas convenir. En effet, le système:

$$\begin{cases} \alpha^i + \alpha^j & = & z_1 \\ f(\alpha^i) + f(\alpha^j) & = & z_2 \end{cases}$$

implique $z_2 = \beta z_1$ et si cette égalité est vérifiée par z_1 et z_2, il équivaut à l'une quelconque de ses équations et ne suffit pas à déterminer les valeurs de i et de j.

• $f(x) = \beta x^2$ ne peut pas convenir non plus. Le système:

$$\left\{ \begin{array}{rcl} \alpha^i + \alpha^j & = & z_1 \\ f(\alpha^i) + f(\alpha^j) & = & z_2 \end{array} \right.$$

équivalent à:

$$\left\{ \begin{array}{rcl} \alpha^i + \alpha^j & = & z_1 \\ \beta\alpha^{2i} + \beta\alpha^{2j} & = & z_2 \end{array} \right.$$

implique $z_2 = \beta(\alpha^{2i} + \alpha^{2j}) = \beta(\alpha^i + \alpha^j)^2 = \beta {z_1}^2$ et si cette égalité est vérifiée par z_1 et z_2, il équivaut à l'une quelconque de ses équations.

• Montrons maintenant que $f(x) = x^3$ convient; le système s'écrit:

$$\left\{ \begin{array}{rcl} \alpha^i + \alpha^j & = & z_1 \\ \alpha^{3i} + \alpha^{3j} & = & z_2 \end{array} \right. .$$

On a: $(\alpha^i + \alpha^j)^3 = \alpha^{3i} + \alpha^{3j} + \alpha^{2i+j} + \alpha^{i+2j} = \alpha^{3i} + \alpha^{3j} + \alpha^{i+j}(\alpha^i + \alpha^j)$. Le sytème équivaut donc à:

$$\left\{ \begin{array}{rcl} \alpha^i + \alpha^j & = & z_1 \\ (\alpha^i + \alpha^j)^3 + \alpha^{i+j}(\alpha^i + \alpha^j) & = & z_2 \end{array} \right.$$

soit encore à:

$$\left\{ \begin{array}{rcl} \alpha^i + \alpha^j & = & z_1 \\ {z_1}^3 + \alpha^{i+j} z_1 & = & z_2 \end{array} \right. .$$

Puisque $i \neq j$, z_1 est nécessairement non nul et la résolution du système se ramène, en remplaçant α^j par sa valeur en fonction de $x = \alpha^i$, à la résolution de l'équation: $x(z_1 + x) + {z_1}^2 = \dfrac{z_2}{z_1}$, c'est à dire:

$$x^2 + z_1 x + {z_1}^2 + \frac{z_2}{z_1} = 0.$$

On peut montrer que cette équation admet toujours des racines lorsqu'il y a eu 2 erreurs.

Définition 20 *Soit α un élément primitif du corps $GF(2^m)$. Le code BCH 2-correcteur de longueur $n = 2^m - 1$ est le code dont une matrice de parité est de la forme:*

$$H'_m = \left[\begin{array}{cccc} \alpha & \alpha^2 & \ldots & \alpha^n \\ \alpha^3 & \alpha^6 & \ldots & \alpha^{3n} \end{array} \right].$$

Ce code est un code linéaire $[n,n-2m,5]$.

Exercice 29 *Déduire de l'exercice 28 la matrice de parité du code BCH* 2*-correcteur de longueur* 15*.*

Remarque 11 *Il est possible de généraliser cette notion de code BCH dans deux directions:*
- de la même façon qu'on a ajouté des lignes à la matrice de parité du code de Hamming pour obtenir celle du code BCH 2*-correcteur, on peut ajouter des lignes à celle-ci pour avoir la matrice de parité du code BCH* 3*-correcteur, et ainsi de suite. On obtient la matrice:*

$$\begin{bmatrix} \alpha & \alpha^2 & \dots & \alpha^n \\ \alpha^3 & \alpha^6 & \dots & \alpha^{3n} \\ \alpha^5 & \alpha^{10} & \dots & \alpha^{5n} \\ \dots & \dots & \dots & \dots \\ \alpha^{2k-1} & \alpha^{2(2k-1)} & \dots & \alpha^{n(2k-1)} \end{bmatrix},$$

(à laquelle il faut, en toute rigueur, supprimer certaines lignes, s'il en existe qui peuvent s'obtenir par combinaisons linéaires des autres).
- on peut adapter la même construction à des codes non binaires: le corps $GF(2) = \{0,1\}$ *devient alors, par exemple, le corps* $GF(p) = \{0,1,\dots,p-1\}$ *des entiers modulo* p*, où* p *est un nombre premier, ou encore le corps* $GF(2^r)$ *tel qu'il a été construit à la page 63.*

7.2 Codes cycliques: généralités

Les codes de Hamming tels qu'ils sont décrits dans la remarque 11 ont une propriété qui semble à première vue mineure et qui va s'avérer fondamentale: ils sont cycliques (voir ci-dessous).

Dans cette section, p désignera toujours un nombre premier; $q = p^r$ sera l'ordre du corps de base K du code considéré. Etant donné un code C linéaire de longueur n sur $GF(q)$, on supposera que $pgcd(n,q) = 1$. La raison pour laquelle on fait cette hypothèse sera explicitée plus loin.

Les codes cycliques (CRC pour Cyclic Redundant Codes) sont les codes classiques les plus importants de la théorie. Si l'on se donne une longueur n et un corps de base, on dispose d'un choix assez large de codes cycliques, determinés alors selon leur capacité de correction et/ou leur dimension. Les codes cycliques contiennent les codes les plus "performants" au niveau des

applications, les codes de BOSE-CHAUDHURY-HOCQUENGHEM, pour lesquels on dispose de bons algorithmes de décodage. Enfin, ces codes sont importants sur le plan théorique.

Remarque 12 *Avant d'introduire et d'étudier la notion de code cyclique, il est important de faire la remarque suivante: jusqu'à maintenant, si nous avons considéré des codes dont les corps de base étaient des corps finis généraux, c'était dans un souci de généralité. Mais on pouvait très bien se restreindre au seuls codes binaires et ignorer tout des corps finis généraux.*
Il va maintenant en être tout autrement: même en se restreignant aux codes binaires, l'étude des codes cycliques va nous amener à nous plonger dans les extensions du corps de base.
Un code cyclique est un code linéaire défini par une propriété de son groupe d'automorphismes.

Définition 21 *Soit C un code linéaire de longueur n; soit σ une permutation de $\{1,\ldots,n\}$. On identifie σ à une permutation des coordonnées des mots de C.*

Alors, σ est un automorphisme du code C si et seulement si $\sigma(C) = C$.

On vérifie facilement que l'ensemble des automorphismes d'un code (linéaire ou non), muni de la composition des applications, est un groupe.
Cette définition est une version restrictive des automorphismes d'un code. La définition générale prend en compte toutes les transformations qui conservent la distance de HAMMING; ces transformations sont dites *isométriques*.

Définition 22 *Un code linéaire C de longueur n sur K est un code cyclique si et seulement si son groupe d'automorphismes contient le shift, c'est à dire s'il a la propriété:*

$$(c_0,\ \ldots,\ c_{n-1}) \in C \quad \Longleftrightarrow \quad (c_{n-1},\ c_0,\ \ldots,\ c_{n-2}) \in C \ .$$

On dit que le code C est invariant par le shift.

Nous allons étudier maintenant une définition équivalente des codes cycliques qui va nous permettre de les construire effectivement. A tout mot de longueur n sur K: $c = (c_0,\ \ldots,\ c_{n-1})$, on peut faire correspondre le polynôme $f(X) = \sum_{i=0}^{n-1} c_i X^i$. Le polynôme associé selon le même principe au "shifté" de c s'obtient en multipliant $f(X)$ par X puis en remplaçant X^n par 1. Cela signifie que la multiplication par X a été effectuée non pas

dans l'algèbre des polynômes, mais dans l'algèbre quotient $R[X] = \frac{K[X]}{(X^n-1)}$, i.e. l'ensemble des classes d'équivalence relatives à la relation d'équivalence: $f(X) \sim g(X) \Longleftrightarrow X^n - 1 \,|\, g(X) - f(X)$.
Dans la suite, on identifiera tout mot de longueur n à un polynôme de degré au plus $n-1$, et ce polynôme à l'élément de l'algèbre $R[X]$ auquel il appartient.

Théorème 4 *On suppose que $pgcd(n,q) = 1$. Un code cyclique de longueur n sur $K = GF(q)$ est un idéal de l'algèbre quotient:*

$$R[X] = \frac{K[X]}{(X^n - 1)} \quad .$$

Tout idéal de cette algèbre est principal. Lorsque $n = q^m - 1$ les codes sont dits primitifs.

Preuve:
On a vu que $R[X]$ est l'algèbre des polynômes à une indéterminée, de degré inférieur à n à coefficients dans K, munie des opérations + et $\times$ modulo $X^n - 1$.
L'algèbre $R[X]$ est commutative.
Le "shift" correspond à la multiplication par X. Un idéal (i.e. un sous-espace de $R[X]$ stable par multiplication par les éléments de $R[X]$) est un sous-espace invariant par le shift.
Il suffit donc de montrer que l'algèbre $R[X]$ est principale, c'est à dire que tout idéal est principal (i.e. est l'ensemble des produits d'un polynôme donné et de tous les éléments de l'algèbre). Montrons plus précisément que tout idéal I est engendré par le polynôme:

$$g(X) = \sum_{i=0}^{r} g_i X^i, \;\; g \in I \;, r = \min\{\; d^\circ f \mid f \in I^*\} \;, g_r = 1$$

En effet, soit $c(X)$ un élément de l'idéal; la division euclidienne de $c(X)$ par $g(X)$ donne: $c(X) = g(X)h(X) + r(X)$, avec $\mathrm{d}^\circ r < \mathrm{d}^\circ g$, ce qui implique $r(X) = 0$ puisque $r(X)$ est élément de l'idéal et que son degré est inférieur au degré minimal. Ce polynôme normalisé de degré minimal contenu dans l'idéal est donc un polynôme générateur du code cyclique. $\diamond$

Remarque 13 *D'après ce qui précède, le polynôme $g(X)$ est unique. Par contre, il existe en général plusieurs polynômes générateurs du code.*
A l'avenir, quand on parlera du *polynôme générateur du code, il s'agira du polynôme normalisé de degré minimal.*

Exercice 30 *Montrer que si $g(X)$ est le polynôme normalisé de plus petit degré contenu dans un idéal de $R[X]$, alors il divise $X^n - 1$ (en tant que polynôme de degré inférieur à n).*

Ce résultat se démontre en faisant la division euclidienne de $X^n - 1$ par $g(X)$.

Exercice 31 *Montrer que si $f(X)$ est un polynôme quelconque et si C est le code cyclique égal à l'ensemble des multiples de $f(X)$ dans l'algèbre $R[X]$, alors le polynôme générateur de C est égal au PGCD de $f(X)$ et $X^n - 1$.*

7.3 Dimension et matrice génératrice d'un code cyclique

Proposition 11 *Soit C un code cyclique $[n,k]$ sur $GF(q)$, où k est la dimension de C. Soit $g(X)$ le polynôme générateur de C et t le degré de $g(X)$. Alors* **k=n-t**.

Preuve: On considère le *mot* g: $g = (g_0, \ldots, g_t)$. Et l'on construit la matrice suivante:

$$G = \begin{bmatrix} g_0 & g_1 & \ldots & & g_t & 0 & \ldots & 0 \\ 0 & g_0 & \ldots & & g_{t-1} & g_t & & 0 \\ \vdots & & & & & & & \vdots \\ 0 & & \ldots & 0 & g_0 & \ldots & & g_t \end{bmatrix} \quad i.e. \quad \begin{matrix} g \\ Xg \\ \vdots \\ X^{n-t-1}g \end{matrix}$$

Il est clair que G est de rang $n - t$.
Or, la matrice G est la matrice génératrice de C. En effet, les éléments de C sont les polynômes de degrés inférieurs à n qui sont divisibles par $g(X)$ (dans $K[X]$, voir remarque 13, page 68). On a alors:

$$\exists\, h(X) \;;\; d^\circ h + d^\circ g < n \quad \text{et} \quad gh = f \quad \Longleftrightarrow \quad f(X) = \sum_{i=0}^{n-t-1} \lambda_i\, X^i\, g(X) \;.$$

$\diamond$

7.4 Dual d'un code cyclique

Exercice 32 *Montrer que deux codes linéaires orthogonaux ont le même groupe d'automorphismes.*

Le dual d'un code cyclique est donc un code cyclique.

Définition 23 *Soit C un code cyclique $[n,k]$ sur $GF(q)$, engendré par $g(X)$. On appelle polynôme de contrôle de C le polynôme*

$$h(X) \in R[X] \ : \qquad X^n - 1 \ = \ h(X)\ g(X)\ ,$$

Remarque 14 $d^\circ h \ = \ n \ - \ d^\circ g \ = \ dim\ C \ = \ k.$

Proposition 12 *On note $h(X) = \sum_{i=0}^{k} h_i X^i$, le polynôme de contrôle de C. Alors la matrice suivante est une matrice de contrôle pour le code C:*

$$H = \begin{bmatrix} 0 & \dots & 0 & h_k & \dots & h_1 & h_0 \\ 0 & \dots & h_k & h_{k-1} & \dots & h_0 & 0 \\ \vdots & & & & & & \vdots \\ h_k & \dots & h_0 & 0 & \dots & & 0 \end{bmatrix}$$

Preuve: $f(X)$ appartient à C si et seulement si $X^n - 1$ divise $h(X)f(X)$ (par multiplication par $h(X)$), i.e. si et seulement si $h(X)f(X) = 0 \mod [X^n - 1]$. On a:

$$h(X)f(X) \mod [X^n - 1] = \sum_{i=0}^{n-1} (\sum_{j=0}^{n-1} f_j h_{i-j}) X^i$$

(l'indice $i-j$ étant pris modulo n). Donc $f(X)$ appartient à C si et seulement si pour tout $i = 0, \cdots, n-1$ on a $\sum_{j=0}^{n-1} f_j h_{i-j}$. Cela implique que $f(X)$ est orthogonal à chaque ligne de H.
C est donc inclus dans le code C' admettant H pour matrice de parité. On a $k = n - t$. Donc C' a même dimension k que C, ce qui implique que $C = C'$. ◇

Proposition 13 *Le polynôme générateur du code $C^\perp$ est :*

$$\tilde{h}(X) = \frac{X^k}{h_0} h(X^{-1}) \quad , k = n - t \quad , t = d^\circ g \quad .$$

Cela se voit directement sur la matrice génératrice H du code $C^\perp$. On a divisé par h_0 pour avoir un polynôme normalisé. ◇

7.5 Encodage

La dimension du code étant $n-t$, les mots sources sont des mots de longueur $n-t$. Soit donc un mot-source $(u_0,\ldots,u_{n-t-1})$. On représente ce mot par le polynôme $U(X) = u_0 + u_1X + \ldots + u_{n-t-1}X^{n-t-1}$.
Une première façon d'encoder est de le faire par la matrice génératrice ci-dessus. Le mot de code correspondant au mot-source $(u_0,\ldots,u_{n-t-1})$ est égal à:

$$(u_0,\ldots,u_{n-t-1})\,G$$

Exercice 33 *Montrer que le polynôme représentant le mot de code*

$$(u_0,\ldots,u_{n-t-1})\,G$$

s'obtient en calculant le produit des deux polynômes $U(X)$ et $g(X)$.
L'encodage correspond donc très simplement à une multiplication de polynômes.
Une deuxième façon d'encoder est de calculer le polynôme $X^t\,U(X)$ et de lui retrancher le reste de sa division par $g(X)$. Le polynôme obtenu est bien de la forme $q(X)\,g(X)$, et à chaque mot-source correspond un mot de code différent:
Exercice 34 *Vérifier que si $X^t\,U(X) = q(X)\,g(X) + R(X)$ et $X^t\,V(X) = q(X)\,g(X) + R'(X)$ avec $U(X)$ et $V(X)$ de degrés inférieurs à $n-t$, $R(X)$ et $R'(X)$ de degrés inférieurs à t alors $U(X) = V(X)$.*

7.6 Construction des codes cycliques

D'après ce qui précède, déterminer tous les codes cycliques sur K de longueur n est équivalent à déterminer tous les diviseurs $g(X)$ de X^n-1 dans $K[X]$.
Exercice 35 *Montrer que, puisque $pgcd(n,q) = 1$, X^n-1 n'a pas de facteurs multiples (calculer sa dérivée) et qu'il en est donc de même de $g(X)$.*

. Il est possible de factoriser X^n-1 en produit de polynômes irréductibles sur K:
rappelons que si $G = F_{q^m}$ (corps fini d'ordre q^m) est l'extension de degré m du corps K, alors le polynôme $X^{q^m-1}-1$ se scinde de la façon suivante:

$$X^{q^m-1} - 1 = \prod_{\beta\in\mathbf{G}^*} (X-\beta).$$

Supposons que l'on ait trouvé un entier m tel que n divise $q^m - 1$. Alors le polynôme $X^n - 1$ divise $X^{q^m-1} - 1$.

Exercice 36 *Montrer que dans un corps quelconque,* $X^n - 1$ *divise* $X^{n'} - 1$ *si et seulement si* n *divise* n'.

$X^n - 1$ se scinde donc sur G. Il existe ainsi un ensemble T tel que:

$$X^n \ - \ 1 = \prod_{\beta \in T} (X - \beta) \quad , \ T \subset \mathbf{G}^* \ .$$

Or, un tel entier m existe toujours, d'après le théorème d'Euler-Fermat:

$$(n,q) = 1 \quad \Longrightarrow \quad \exists \lambda \ : \ q^\lambda \equiv 1 \mod n \ .$$

L'ordre multiplicatif m *de* q *modulo* n est par définition le plus petit de ces nombres λ. L'extension $G = F_{q^m}$ de degré m de K s'appelle le *corps de décomposition* de $X^n - 1$.

Le code I, engendré par $g(X)$ est défini par un ensemble T_g qui vérifie: $T_g \subset T \subset \mathbf{G}^*$:

$$I = < g(X) > \ : \ g(X) = \prod_{\beta \in T_g} (X - \beta) \ .$$

L'ensemble T_g est l'ensemble des **zéros** du code I. *Un polynôme* $c(X)$*, élément de l'algèbre* $A[X]$*, appartient au code si et seulement si tous les éléments de* T_g *sont racines de ce polynôme.*

Remarque 15 *Contrairement à certains des résultats qui précèdent, cette condition est indépendante du choix du représentant de l'élément du code dans la classe correspondante de* $A[X]$.

. Le code ayant ses symboles dans le corps K d'ordre q, il nous reste à déterminer à quelle condition l'ensemble des zéros T_g est tel que $g(X)$ soit un polynôme à coefficients dans K.

Proposition 14 *Soit* $g(X) = \prod_{\beta \in T_g} (X - \beta)$ *où* T_g *est un sous-ensemble du corps* $G = F_{q^m}$.

Alors, $g(X)$ *est à coefficients dans* $K = F_q$ *si et seulement si* T_g *est stable par l'automorphisme de Frobenius* $\beta \to \beta^q$.

Preuve: Posons $g(X) = \sum_{i=0}^{n-1} a_i \, X^i$, $a_i \in G$.

Rappelons que K est l'ensemble des éléments x de G tels que $x^q = x$.

Si tous les coefficients a_i sont dans K, alors on a pour toute racine β de $g(X)$:

$$g(\beta^q) = \sum_{i=0}^{n-1} a_i \, \beta^{qi} = \sum_{i=0}^{n-1} {a_i}^q \, \beta^{qi} = (g(\beta))^q = 0$$

et T_g est donc bien stable par l'automorphisme de corps $\beta \to \beta^q$.
Réciproquement, si T_g est stable par l'automorphisme de corps $\beta \to \beta^q$, alors on a:

$$(g(X))^q = \prod_{\beta \in T_g} (X^q - \beta^q) = \prod_{\beta \in T_g} (X^q - \beta) = g(X^q).$$

Cela signifie que les deux polynômes $\sum_{i=0}^{n-1} (a_i)^q X^{qi}$ et $\sum_{i=0}^{n-1} a_i X^{qi}$ sont égaux et donc que les coefficients a_i sont tels que ${a_i}^q = a_i$. $\diamond$

Rappelons que tout corps fini est cyclique: il existe un élément α du corps tel que les éléments non nuls de celui-ci soient tous de la forme α^i.
α s'appelle un élément primitif du corps.
Soit donc α un *élément primitif* de G. Cet élément étant une racine du polynôme irréductible qui intervient dans la définition du corps, on parlera aussi de *racine primitive*. On a:

$$G^* = \{\alpha^i, i = 0, \ldots, q^m - 2\}$$

et, d'après la proposition 14, $g(X)$ est à coefficients dans K si et seulement si T_g est de la forme $\{\alpha^i, i \in E_g\}$, où E_g est stable par multiplication par q modulo $q^m - 1$, i.e. est une réunion de classes cyclotomiques de q modulo $q^m - 1$.
Définition 24 *Soit s un entier. La classe cyclotomique de q modulo $q^m - 1$ contenant s est l'ensemble des $q^i\ s \mod q^m - 1$ quand i varie dans les entiers.*

Exercice 37 *Soit $S = [0, q^m - 2] = \{0, \ldots, q^m - 2\}$. Vérifier que la relation $\mathcal{R}$ sur S, définie par*

$$\forall\ s,t \in S\ :\ s\ \mathcal{R}\ t \quad \Leftrightarrow \quad \exists\ i\ :\ q^i\ s \equiv t \mod q^m - 1$$

est une relation d'equivalence, et que les classes de $\mathcal{R}$ sont les classes cyclotomiques *de q modulo $q^m - 1$.*
Montrer que le cardinal d'une classe est un diviseur de m.

On en déduit que $g(X)$ est à coefficients dans K si et seulement si il est le produit de polynômes de la forme $\prod_{i \in E_j} (X - \alpha^i)$, où les E_j sont des classes cyclotomiques distinctes (donc disjointes) de q modulo $q^m - 1$.
Définition 25 *Soit $\beta \in \mathbf{G}^*$. Le polynôme minimal de β sur K, noté $M_\beta(X)$, est le polynôme normalisé de plus petit degré, sur K vérifiant $M_\beta(\beta) = 0$.*

On note $cl(s)$, $s \in S$, la classe cyclotomique de q modulo $q^m - 1$ contenant s.
Exercice 38 *Montrer qu'un polynôme est le minimal d'un élément si et seulement si il est de la forme* $\prod_{i \in cl(s)}(X - \alpha^i)$.
En déduire que $g(X)$ *est à coefficients dans* K *si et seulement si il est le produit de polynômes minimaux distincts.*

Exercice 39 *Montrer que:*

1. M_β *est irréductible sur* $K = F_q$.
2. $f \in K[X]$ *et* $f(\beta) = 0 \quad \Longrightarrow \quad M_\beta$ *divise* f .
3. M_β *divise* $(X^{q^m-1} - 1)$.
4. *Le degré de* M_β *est inférieur ou égal à* m.
5. *Les éléments* β, β^q, β^{q^2}, ... *ont le même polynôme minimal.*

En conclusion:

Les codes cycliques sont les idéaux principaux dont les polynômes générateurs sont de la forme:

$$g(X) = \prod_{k \in D}(X - \alpha^k) \;=\; \prod_i M_{\alpha^{s_i}}(X)$$

où D *est un sous-ensemble de* S *qui est une réunion de classes cyclotomiques et où les zéros* α^k, $k \in D$, *sont des racines du polynôme* $X^n - 1$.
Exercice 40 *Montrer:*

$$\forall k \in D \;:\; k = k'n' \quad où \quad n'n = q^m - 1 \;.$$

Tous les éléments de D *étant divisibles par* n', *le plus simple est de poser* $\beta = \alpha^{n'}$ *(*β *est une racine primitive* n*-ième de l'unité) et de considérer les classes cyclotomiques de* q *modulo* n.
D_1 *étant alors la réunion de telles classes, on peut définir:*

$$g(X) = \prod_{k \in D_1}(X - \beta^k) \;.$$

Remarque 16 *Tout polynôme ainsi construit est le polynôme normalisé de degré minimal du code cyclique qu'il engendre. Il est donc unique.*
Il y a donc autant de codes cycliques que de façon de choisir $g(X)$.
Si k *est le nombre de classes cyclotomiques de* q *modulo* n, *il y a donc* 2^k *codes cycliques distincts. Disons* $2^k - 2$ *si l'on exclut le code réduit au mot nul (qui correspond à* $g(X) = X^n - 1$*) et le code égal à* K^n *tout entier (qui correspond à* $g(X) = 1$*).*

`Exemple :` $q = 2$, $n = 5$. Il est clair que $X^5 - 1$ divise $X^{15} - 1$, l'ordre multiplicatif de 2 modulo 5 est égal à 4 et donc F_{16} est le corps de décomposition de $X^5 - 1$. Soit α une racine primitive de F_{16} et $\beta = \alpha^3$.
Les racines de $X^5 - 1$ sont: $\{ 1, \beta, \beta^2, \beta^3, \beta^4 \}$.
Les classes cyclotomiques de 2 modulo 5 étant $\{0\}$ et $\{1,2,3,4\}$, il n'y a que deux possibilités pour un polynôme générateur $g(X) \in F_2[X]$:

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

7.7 Zéros du dual

On peut déduire directement les **zéros** du code $C^\perp$ de ceux de C: soit $F = F_{q^m}$, le corps de décomposition de $X^n - 1$, α une racine primitive de F_{q^m} et

$$n'n = q^m - 1 \ , \ \gamma = \alpha^{n'} \qquad (\Rightarrow \quad \gamma^n = 1 \ ) \ .$$

Exercice 41 *Montrer:*

$$g(X) = \prod_{j \in J} (X - \gamma^j) \quad \Longrightarrow \quad \tilde{h}(X) = \prod_{u \notin J} (X - \gamma^{n-u}) \ .$$

Les zéros de $C^\perp$ sont donc les inverses des non-zéros de C.

`Exemple:` longueur $n = 15$, $q = 2$

$$\begin{aligned} X^{15} - 1 \ &= \ \left(X^4 + X^3 + 1\right)\left(X^4 + X + 1\right)\left(X^4 + X^3 + X^2 + X + 1\right) \\ &\quad (X + 1)\left(X^2 + X + 1\right). \end{aligned}$$

Le polynôme $X^4 + X + 1$ est primitif (l'une de ses racines est un élément primitif de F_{16}). Soit α une racine de ce polynôme et C le code engendré par :

$$g(X) = (X - \alpha)(X - \alpha^2)(X - \alpha^4)(X - \alpha^8) = X^4 + X + 1 \ .$$

Alors $dim\ C = 15 - d^\circ g = 11$. Le code C est (15,11,3). Son polynôme de contrôle est :

$$h(X) = \frac{X^{15} - 1}{X^4 + X + 1} = X^{11} + X^8 + X^7 + X^5 + X^3 + X^2 + X + 1 \ .$$

On obtient aussi le générateur du dual de C:

$$\tilde{h}(X) = \sum_{i=0}^{11} h_{11-i} X^i = X^{11} + X^{10} + X^9 + X^8 + X^6 + X^4 + X^3 + 1 \; .$$

En représentant la matrice génératrice (du dual) correspondant à ce polynôme générateur, on retrouve la matrice de contrôle qui définit le code de HAMMING $[2^m - 1, 2^m - m - 1, 3]$ pour $m = 4$:

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

Chapitre 8

Principaux codes cycliques

8.1 Les codes de Bose-Chaudury-Hocquenghem (codes BCH)

Les codes BCH sont les codes cycliques de plus grande dimension pour une capacité de correction donnée; cette capacité, est assurée par une borne sur leur distance minimale, qu'on appelle *la borne BCH*. Celle-ci assure que la distance minimale d'un code BCH est minorée par une valeur, appelée la *distance construite* du code. Notons que la distance minimale est en fait souvent strictement supérieure à la distance construite.

8.1.1 La borne BCH

Soit C un code cyclique de longueur n; soit $G = F_{q^m}$ le corps de décomposition de $X^n - 1$. On désigne par α une racine primitive de G. Le polynôme générateur de C est:

$$g(X) = \prod_{j \in J} (X - \beta^j) \quad , \; n'n = q^m - 1 \; , \; \beta = \alpha^{n'} \; .$$

β est une racine n-ième primitive de l'unité.
L'ensemble J est **l'ensemble de définition** du code cyclique C. Rappelons qu'un élément de l'algèbre $A[X]$ appartient au code si et seulement si tous les β^j,$j \in J$ sont racines de ce polynôme.

Théorème 5 *Si l'ensemble J comporte $d-1$ valeurs consécutives, alors le*

code C a une distance minimale δ supérieure ou égale à d. La plus grande valeur de d obtenue par ce procédé est la borne-BCH du code C.

Pour calculer la borne BCH du code, on calcule donc la longueur de la plus grande chaîne d'éléments consécutifs (modulo n) de l'ensemble de définition et on l'augmente de 1.
Preuve: Soit $\{\beta^{l+j}, j \in [1,d-1]\}$ l'ensemble des zéros du code. Soit $x \in C$. Alors : $\forall\ j \in [1,d-1]\ :\ \sum_{i=0}^{n-1} x_i(\beta^{l+j})^i = 0$.
Les coordonnées de x sont donc solutions d'un système dont la matrice est:

$$\begin{bmatrix} 1 & \beta^l\beta & \beta^{2l}\beta^2 & \dots & \beta^{(n-1)l}\beta^{n-1} \\ 1 & \beta^l\beta^2 & \beta^{2l}(\beta^2)^2 & \dots & \beta^{(n-1)l}(\beta^2)^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \beta^l\beta^{d-1} & \beta^{2l}(\beta^{d-1})^2 & \dots & \beta^{(n-1)l}(\beta^{d-1})^{n-1} \end{bmatrix} .$$

Si on choisit $d-1$ colonnes, $c_1,\dots,c_t$ $(t = d-1)$, on obtient un système $(d-1)\times(d-1)$:

$$\begin{bmatrix} \beta^{c_1 l}\beta^{c_1} & \beta^{c_2 l}\beta^{c_2} & \dots & \beta^{c_t l}\beta^{c_t} \\ \vdots & \vdots & & \vdots \\ \beta^{c_1 l}(\beta^t)^{c_1} & \beta^{c_2 l}(\beta^t)^{c_2} & \dots & \beta^{c_t l}(\beta^t)^{c_t} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{c_t} \end{bmatrix} = 0 \ .$$

Le déterminant de ce système est un déterminant de Vandermonde:

$$\begin{vmatrix} \beta^{c_1 l}\beta^{c_1} & \beta^{c_2 l}\beta^{c_2} & \dots & \beta^{c_t l}\beta^{c_t} \\ \vdots & \vdots & & \vdots \\ \beta^{c_1 l}(\beta^t)^{c_1} & \beta^{c_2 l}(\beta^t)^{c_2} & \dots & \beta^{c_t l}(\beta^t)^{c_t} \end{vmatrix} = \beta^N \prod_{i<j} (\beta^{c_j} - \beta^{c_i}) .$$

avec $N = (l+1)\sum_{i=1}^{t} c_i$.

Le système n'est donc vérifié que si $x_{c_1} = x_{c_2} = \dots = x_{c_t} = 0$.
Cela signifie que le code C ne peut pas contenir de mots de code de poids au plus $d-1$, et donc que la distance minimale du code est au moins d. $\diamond$

Il y eut de nombreux travaux dont le but était l'**amélioration de la borne-BCH**. Plus précisément on cherche à mettre en évidence des propriétés combinatoires de l'ensemble de définition telles que certains outils algébriques puissent s'appliquer, de la même façon que pour la borne BCH. On a ainsi déterminé plusieurs bornes, chacune généralisant la précédente:

– La borne de Hartmann-Tzeng (1972)

– La borne de Roos (1983)
– La borne de Van-lint et Wilson (1986).

8.1.2 Définition des codes BCH

Ces codes ont été définis en 1959/60. Il sont à l'origine de la définition générale des codes cycliques[1]

Définition 26 *Un code cyclique C de longueur n sur F_q est un code BCH de distance construite d si son polynôme générateur a la forme suivante:*

$$g(X) = \prod_{i \in I} M_{\beta^i}(X) \ ,$$

où β est une racine nième de l'unité et où la réunion des classes cyclotomiques des éléments de I contient tous les entiers entre $l+1$ et $l+d-1$. Si $l=1$, il s'agit d'un code-BCH au sens strict.

Remarque 17 *Le code de Hamming est le code BCH au sens strict de distance construite 3 et de longueur 2^m-1 sur F_2.*
En effet, un mot $f(X) = f_0 + \ldots + f_{n-1}X^{n-1}$ appartient à ce code BCH si et seulement si $f(\alpha)=0$, puisque $f(\alpha^2) = (f(\alpha))^2$.
$f(\alpha)$ est égal à $\sum_{i=0}^{n-1} c_i\alpha^i$. Donc une matrice de parité de ce code est la matrice dont les colonnes sont les coordonnées de $1,\alpha,\ldots,\alpha^{n-1}$ dans une base quelconque de F_{2^m} sur F_2. Cette matrice se note de façon abrégée sous la forme:

$$\begin{bmatrix} 1 & \alpha & \ldots & \alpha^{2^m-2} \end{bmatrix}$$

On retrouve bien la matrice de parité du code de Hamming, puisque α^i décrit tout l'ensemble des éléments non nuls de F_{2^m}.
De même, le code BCH au sens strict de distance construite 5 et de longueur 2^m-1 sur F_2 admet pour matrice de parité, en notation abrégée:

$$\begin{bmatrix} 1 & \alpha & \ldots & \alpha^{2^m-2} \\ 1 & \alpha^3 & \ldots & \alpha^{3(2^m-2)} \end{bmatrix}$$

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

R.C. Bose, D.K. Ray-Chauduri *On a class of error correcting binary group codes*, Inform. and Control, March 1960.

R.C. Bose, D.K. Ray-Chauduri *Further results on error correcting binary group codes*, Inform. and Control, Sept 1960.

COMMENTAIRES :

- Par construction, la distance minimale d'un code BCH est supérieure ou égale à sa distance construite. Elle peut lui être strictement supérieure!
- La borne BCH d est une "bonne" borne pour les codes BCH, surtout lorsqu'ils sont primitifs : pour une longueur du type $2^m - 1$, les résultats numériques dont on dispose actuellement donnent une distance minimale $d \leq \delta \leq d + 4$.
- Pour n donné, il y a autant de codes BCH *au sens strict* que de classes cyclotomiques de q modulo n.

`Exemple 1:` *Construction d'un code BCH de longueur* 9 *et de distance construite* 4 *sur* F_2.

1. Le corps de décomposition de $X^9 - 1$ est F_{64} :

$$X^9 - 1 \quad \text{divise} \quad X^{63} - 1 \qquad , \; 63 = 2^6 - 1 \; .$$

2. On cherche $\beta \in F_{64}$, d'ordre 9 (i.e. $\beta^9 = 1$ et $\beta^i \neq 1$ pour $i < 9$).

$$X^9 - 1 = (X^3 - 1)(X^6 + X^3 + 1) \quad \Rightarrow \quad \beta^6 + \beta^3 + 1 = 0 \; .$$

$$\Longrightarrow$$

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

3. Du fait que la distance construite d doit être 4, la seule possibilité pour l'ensemble de définition est qu'il contienne au moins $\{0,1,2\}$. D'où le polynôme générateur

$$\begin{aligned} g(X) \;\; &= \;\; M_{\beta^0}(X) M_{\beta^1}(X) = (X-1)(X^6 + X^3 + 1) \\ &= \;\; X^7 + X^6 + X^4 + X^3 + X + 1 \;\; . \end{aligned}$$

La dimension du code est $n - 7 = 2$, et son ensemble de définition :

$$\{\; 0,1,2,4,8,7,5 \;\} \;\; \text{où} \;\; 7 = 16 \pmod 9,\; 5 = 32 \pmod 9 \;\; .$$

4. La matrice génératrice nous montre la distance "vraie" du code:

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} .$$

Il s'agit d'un code ayant un seul poids non nul, 6.

`Exemple 2:` *Construction des codes BCH de longueur* 15 *(donc primitifs) sur* F_2. Soit α une racine primitive de F_{16}, racine du polynôme irréductible (X^4+X+1). On décompose $X^{15}-1$, en faisant ainsi apparaître les polynômes minimaux $M(\alpha^j)$:

$$\begin{aligned} X^{15}-1 &= (X-1)\underbrace{(X^4+X+1)}_{M(\alpha)}\underbrace{(X^4+X^3+X^2+X+1)}_{M(\alpha^3)} \\ &\quad \underbrace{(X^2+X+1)}_{M(\alpha^5)}\underbrace{(X^4+X^3+1)}_{M(\alpha^7)} . \end{aligned}$$

Il y a quatre codes BCH dont un trivial. On les obtient en ajoutant successivement, dans l'ordre (distance construite croissante), les classes cyclotomiques.

- {1,2,4,8} , BCH (15,11,3).
- + {3,6,9,12} , BCH (15,7,5).
- + {5,10} , BCH (15,5,7).
- + {7,11,13,14 } , BCH (15,1,15) (trivial).

Déterminer la distance minimale "vraie" des codes BCH est un problème de recherche. Les cas où l'on sait déterminer cette distance, selon la distance construite et la longueur, sont limités ; il s'agit de classes très particulières, comme on peut le voir dans le théorème suivant, donné à titre d'exemple. On connait la distance minimale des BCH stricts binaires primitifs de longueur inférieure ou égale à 255 [2].

Théorème 6 *Soit* $n = \delta n_1$. *Alors le code BCH binaire de distance construite* δ *et de longueur* n *a pour distance minimale* δ.

2. D. Augot, P. Charpin & N. Sendrier, *Studying the locator polynomials of minimum weight codewords of BCH-codes*, IEEE Trans. on Info. Theory, vol. 38, n.3, pp 960-973, May 92.

Preuve:
On va exhiber un mot du code de poids δ. On a:

$$X^n - 1 = (X^{n_1} - 1)\underbrace{(X^{n_1(\delta-1)} + X^{n_1(\delta-2)} + \ldots + 1)}_{T(X)} \ .$$

Soit G le corps de décomposition de $X^n - 1$; soit $\beta \in G$ un élément d'ordre n ($\beta^n = 1$). Les racines de $X^n - 1$ sont: $1,\beta,\ldots,\beta^{n-1}$.
Celles de $X^{n_1} - 1$ sont les β^l tels que:

$$l\ n_1 \equiv 0 \mod n$$

soit encore:

$$l \in \{0,\delta,2\delta,\ldots,(n_1 - 1)\delta\} \ .$$

Donc

$$\beta,\ \beta^2,\ldots,\ \beta^{\delta-1} \text{ sont des zéros de } T(X).$$

Ceci induit que $T(X)$ est un mot du code engendré par :

$$g(X) = \prod_{i \in I}(X - \beta^i) \quad , \quad I = \bigcup_{i=1}^{\delta-1} Cl(i) \ ,$$

qui est un code BCH de longueur n et distance construite δ. Or le polynôme $T(X)$ représente un mot de code de poids δ. $\diamond$

`Exemple :` $n = 63 = 3^2 \times 7$. On considère les codes BCH au sens strict de longueur 63. Les distances construites 3,7,9,21 sont les distances minimales des codes concernés.

8.2 Codes de Reed-Solomon (codes RS)

On rappelle que p désigne toujours un nombre premier et que $q = p^r$. Le symbole α désigne une racine primitive du corps fini F_q.

Définition 27 *Un code de Reed-Solomon est un code BCH de longueur $q-1$ sur F_q.*

Le fait que la longueur n du code soit égale à $q-1$ fait que les classes cyclotomiques de q modulo n sont des singletons.

Proposition 15 *Tous les codes de Reed-Solomon sont "Maximum Distance Separable". Leur distance minimale est égale à leur distance construite.*

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

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

puisque les classes cyclotomiques sont des singletons.
On sait que $d \geq \delta$.
De plus, $g(X)$ ne peut pas avoir plus de δ coefficients non nuls, puisqu'il est de degré $\delta - 1$. Donc

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

Finalement : C est un code $[n = q-1, k, \delta = n-k+1]$; c'est un code MDS.
$\diamond$

Proposition 16 *Soit C un code RS, $[n,k,\delta]$ sur F_q. Alors le code $C^{\perp}$, dual de C, est un code RS, $[n, n-k, k+1]$ sur F_q.*

Preuve: Supposons d'abord que C est un code RS *au sens strict*. On sait que le dual d'un code MDS est un code MDS, ce qui nous donne immédiatement les paramètres de $C^{\perp}$. D'autre part, connaissant l'ensemble des zéros de C, on peut déterminer l'ensemble des zéros de son dual $C^{\perp}$; soit J cet ensemble:

$$\begin{aligned} J &= \{\ \alpha^j \mid \alpha^{n-j} \text{ n'est pas un zéro de } C\ \} \\ &= \{\ \alpha^j \mid n-j \in\ [\delta, n]\ \} = \{\ \alpha^j \mid j \in\ [0, n-\delta+1[\ \} \ . \end{aligned}$$

où $\delta = n - k + 1$.
Pour passer maintenant au cas général, il suffit de translater l'ensemble de définition du code C d'un rang l, ce qui revient à translater celui de son dual

d'un rang $-l$. $\diamond$

`Exemple:` Soient $q = 7$, $\delta = 4$ et cette représentation du corps à 7 éléments (5 est racine primitive):

$$F_7 = \left| \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & \alpha^0 & \alpha^4 & \alpha^5 & \alpha^2 & \alpha & \alpha^3 \end{array} \right|$$

Soit

$$g(X) = (X-\alpha)(X-\alpha^2)(X-\alpha^3) = X^3 - X^2 + 4X + 1 \ .$$

Alors le code C, engendré par $< g(X) >$, est un code RS (6,3,4) sur F_7. Le code $C^\perp$ est un code RS (6,3,4). Bien qu'ayant les mêmes paramètres, C et $C^\perp$ sont des codes RS différents. Le code $C^\perp$ est engendré par:

$$\tilde{h}(X) = (X-1)(X-\alpha)(X-\alpha^2) = (X-1)(X-5)(X-4) = X^3-3X^2+X+6$$

La matrice génératrice de C et une matrice systématique déduite:

$$G = \begin{pmatrix} 1 & 4 & -1 & 1 & 0 & 0 \\ 0 & 1 & 4 & -1 & 1 & 0 \\ 0 & 0 & 1 & 4 & -1 & 1 \end{pmatrix} \quad G_{syst} = \begin{pmatrix} 1 & 0 & 0 & 3 & 0 & -3 \\ 0 & 1 & 0 & -3 & 5 & -4 \\ 0 & 0 & 1 & 4 & -1 & 1 \end{pmatrix} .$$

On est parfois amenés à modifier "légèrement" les paramètres d'un code, généralement pour des applications spécifiques. Dans ce cas, on cherche à conserver autant que possible les propriétés du code initial. Toutefois cette démarche a aussi un intérét théorique: définition de nouveaux outils pour étudier le code initial, construction de nouveaux codes. Ainsi en *étendant* les codes RS, on trouve de nouveaux codes MDS.

Définition 28 *Soit C un code linéaire de longueur n tel qu'il existe $c \in C$ vérifiant $\sum_{i=0}^{n-1} c_i \neq 0$, où c_i est le i-ème symbole de c.* **L'extension** *du code C est le code obtenu en adjoignant à chaque mot de C un symbole dit de parité :*

$$\begin{array}{rcl} (c_0,\ldots,c_{n-1}) \in C & \Rightarrow & (\xi,c_0,\ldots,c_{n-1}) \in C_e \\ & & \xi = -\sum_{i=0}^{n-1} c_i \ . \end{array}$$

Il est clair que l'extension d'un code linéaire est un code linéaire.

Proposition 17 *L'extension d'un code RS* $[n,k,d]$ *(au sens strict) est un code MDS, de paramètres* $[n+1,k,d+1]$.

Preuve: Un code cyclique peut être étendu si et seulement si l'élément 1 n'est pas un zéro du code, puisqu'on a:

$$\sum_{i=0}^{n-1} c_i = 0 \qquad \Longleftrightarrow \qquad c(1) = 0 \ .$$

Les éléments de poids minimal du code ne peuvent admettre 1 pour zéro, car ils seraient alors dans un code RS de distance construite $d+1$. Donc le code extension d'un code RS au sens strict a pour distance minimale $d+1$. C'est un code de longueur $n+1$ et de dimension k, identique à celle de C. $\diamond$

Généralisation des codes de Reed-Solomon: On peut montrer (voir exercice corrigé) que tout code de Reed-Solomon au sens strict de distance construite δ est l'ensemble des mots de la forme $(f(\alpha),\cdots,f(\alpha^{q-1}))$, où α est un élément primitif du corps F_q et où $f(X)$ est un polynôme à coefficients dans F_q et de degré au plus $q-\delta-1$. On peut en déduire une généralisation:

Définition 29 (Définition généralisée des codes de Reed-Solomon) *Soit* F_q *un corps fini, on appelle code de Reed-Solomon sur* F_q *l'ensemble de mots de la forme* $(f(a_1),\cdots,f(a_n))$*, où* $n \leq q$ *et où* $a_1,\cdots,a_n$ *sont des éléments distincts de* F_q.

Comme un polynôme de degré au plus d sur un corps ne peut pas admettre plus de d zéros, on en déduit que le code est de distance minimale au moins $n-d$. On retrouve la borne BCH dans ce cas particulier.

8.3 Les codes de Reed-Solomon et le disque compact

8.3.1 Le CD et le CD-ROM

1. Numérisation de l'information

Le signal est traduit en une suite d'octets.
1 seconde = 176400 octets
Le flot est découpé en trames de 24 octets.

2. Encodage par le code CIRC

Sans code correcteur, le disque est inaudible et même destructeur des enceintes acoustiques.
Les erreurs sont massives: une rayure de 1 cm. détruit 33000 bits consécutifs (paquet d'erreurs).
On identifie l'ensemble des 256 octets au corps à 256 éléments.
On dispose ainsi de la possibilité de multiplier les octets entre eux.
Chaque trame est un flot $\alpha_1\,\alpha_2\,\ldots\,\alpha_{24}$ d'éléments du corps.

Encodage:
on utilise un code dérivé du code de Reed-Solomon qui ajoute deux symboles redondants.
On obtient des trames de 26 éléments.
On les entrelace (on lit les premiers symboles des trames, puis les deuxièmes symboles etc...) et on encode à nouveau par un code de Reed-Solomon.
Les trames sont maintenant de 28 éléments. On les entrelace à nouveau.

3. Gravure

Les 1 sont repérés par les changements de niveau sur le disque.

Chapitre 9

Codes non-linéaires; codes $\mathbf{Z}_4$-linéaires

On va introduire dans ce chapitre une classe de codes non linéaires dont les capacités de correction sont strictement meilleures que celles des meilleurs codes linéaires connus de mêmes longueurs et de mêmes cardinaux. Il s'agit des codes de Kerdock. Ce sont des codes de longueur 2^m, qui sont définis comme sous-codes des codes de Reed-Muller. Le code de Kerdock de longueur 2^m, noté $\mathcal{K}_m$, est une réunion de translatés (cosets, en anglais) du code de Reed-Muller d'ordre 1, RM(1,m).
En d'autres termes, il est de la forme $\{f+h, f \in E, h \in \mathrm{RM}(1,m)\}$, où E est un certain ensemble de fonctions booléennes.

Proposition 18 *Soit C un code de longueur 2^m qui soit réunion d'au moins 2 translatés du code RM(1,m). Alors la distance minimale de C est au plus $2^{m-1} - 2^{\frac{m}{2}-1}$.*

Preuve: La distance minimale entre deux translatés $\{f_1+h, h \in \mathrm{RM}(1,m)\}$ et $\{f_2+h, h \in \mathrm{RM}(1,m)\}$ est égale à la distance de f_1+f_2 au code RM(1,m)$\}$. Notons $f = f_1 + f_2$. Pour toute fonction linéaire $l_a(x) = a_1x_1 + \ldots + a_mx_m$, on a $\min(d(f,l_a),d(f,l_a+1)) = 2^{m-1} - \frac{1}{2}|\sum_{x\in F_2^m}(-1)^{f(x)+l_a(x)}|$. On en déduit $d(f,R(1,m)) = 2^{m-1} - \frac{1}{2}\max_{a\in F_2^m}|\sum_{x\in F_2^m}(-1)^{f(x)+l_a(x)}|$. On montre que $\sum_{a\in F_2^m}\left(\sum_{x\in F_2^m}(-1)^{f(x)+l_a(x)}\right)^2 = 2^{2m}$ (exercice!). On en déduit que la moyenne arithmétique de $\left(\sum_{x\in F_2^m}(-1)^{f(x)+l_a(x)}\right)^2$ est égale à 2^m et donc que $\max_{a\in F_2^m}|\sum_{x\in F_2^m}(-1)^{f(x)+l_a(x)}| \geq 2^{m/2}$. Cela implique que $d(f,R(1,m)) \leq 2^{m-1} - 2^{\frac{m}{2}-1}$. $\diamond$

D'après ce qui précède, la distance d'une fonction booléenne à $RM(1,m)$ est au plus égale à $2^{m-1} - 2^{\frac{m}{2}-1}$. Notons que cette valeur ne peut être atteinte que si m est pair. La fonction est dite *courbe* si elle atteint ce maximum.

Lorsqu'un code est non linéaire, sa distance minimale peut être différente de son poids minimal. Ce ne sera pas le cas des codes de Kerdock, car ces codes sont *distance-invariants*.

Définition 30 *Un code C est dit distance-invariant si, pour tout mot a du code, la distribution de poids du translaté $a + C$ est identique à celle du code C lui-même.*

Tout code linéaire est distance-invariant, puisque pour un tel code C, on a $a + C = C$. Mais la réciproque est fausse: les codes de Kerdock seront des contre-exemples de cette réciproque.

9.1 Les codes de Kerdock

Le code $\mathcal{K}_m$ va avoir pour distance minimale $2^{m-1} - 2^{\frac{m}{2}-1}$ (il ne peut donc être défini que pour m pair). Donc, d'après la proposition 18, pour tout couple (f_1,f_2) de fonctions appartenant à deux cosets distincts de $RM(1,m)$ inclus dans $\mathcal{K}_m$, la distance de $f_1 + f_2$ à RM$(1,m)$ doit être égale à $2^{m-1} - 2^{\frac{m}{2}-1}$. Si $f_1 + f_2$ n'est pas affine, alors elle doit donc être courbe.

Il existe diverses définitions de ce code (voir [3]). La plus simple suppose une identification entre F_2^m et $F_{2^{m-1}} \times F_2$ et définit les éléments du code en tant que fonctions booléennes $f(x,\epsilon), x \in F_{2^{m-1}}, \epsilon \in F_2$.

On note tr la fonction trace sur $F_{2^{m-1}}$: $tr(x) = x + x^2 + x^{2^2} + \ldots + x^{2^{m-2}}$. Rappelons que cette fonction est une forme linéaire sur $F_{2^{m-1}}$.

On pose $m = 2t + 2$.

Définition 31 *Soit m un entier pair supérieur ou égal à 4. Le code de Kerdock de longueur 2^m est l'ensemble des fonctions booléennes de la forme $f_u + h$, $h \in RM(1,m)$, où:*

$$f_u(x,\epsilon) = tr\left(\sum_{i=1}^{t}(ux)^{2^i+1}\right) + \epsilon tr(ux),$$

u étant un élément quelconque de $F_{2^{m-1}}$.

Exercice 42 *Calculer le cardinal de $\mathcal{K}_m$.*

Le code de Kerdock est un sous-code du code RM(2,m), en vertu du théorème suivant.

Théorème 7 *Pour tout entier i, notons $\omega_2(i)$ le nombre de "1" dans la décomposition binaire de i. Alors, pour tout $r < m$, RM(r,m) est l'ensemble des fonctions de la forme:*

$$f(x) = \sum_{i \in I} tr\left(a_i\, x^i\right)$$

où pour tout i de I, $a_i \in F_{2^m}$, $0 \leq i \leq 2^m - 1$ et $\omega_2(i) \leq r$.

Il serait maladroit de tenter de démontrer directement ce résultat (tel qu'il est énoncé) car il n'y a pas unicité de la représentation d'une fonction booléenne sous cette forme. Nous allons d'abord montrer le résultat suivant:

Lemme 2 *Soit $\mathcal{P}$ l'espace vectoriel sur F_{2^m} des polynômes à coefficients dans F_{2^m} et dont les degrés sont strictement inférieurs à 2^m. L'application ψ, qui à tout polynôme de $\mathcal{P}$ fait correspondre la fonction polynôme associée, est une bijection de $\mathcal{P}$ sur l'ensemble des fonctions de F_{2^m} dans lui-même (que nous appellerons* fonctions booléennes vectorielles*).*
Soit $\mathcal{P}_r$ le sous-espace vectoriel de $\mathcal{P}$ constitué des polynômes dont tous les exposants sont de 2-poids au plus r. Alors, l'image de $\mathcal{P}_r$ par ψ est l'ensemble des fonctions booléennes vectorielles dont les formes algébriques normales sont de degrés au plus r.

Preuve du lemme:
ψ est clairement une application F_{2^m}-linéaire. Pour montrer qu'elle est bijective, il suffit de montrer qu'elle est injective, puisque ses espaces vectoriels de départ et d'arrivée sont tous deux de dimension 2^m. Or, le noyau de ψ est réduit à $\{0\}$, puisqu'un polynôme non nul de degré au plus $2^m - 1$ ne peut admettre 2^m racines distinctes.
Montrons maintenant par récurrence sur r que l'image de $\mathcal{P}_r$ par ψ est l'ensemble des fonctions booléennes vectorielles dont les formes algébriques normales sont de degrés au plus r.
Cela est trivialement vrai pour $r = 0$.
Admettons ce résultat pour $r - 1$ et montrons le pour r.
Il suffit de montrer une inclusion entre ces deux sous-espaces, puisqu'ils sont de même dimension (le nombre des entiers de $[0,2^m - 1]$ qui sont de 2-poids au plus r est bien égal au nombre des monômes de degrés au plus égaux à r).
Pour montrer que l'image d'un élément $P(X)$ quelconque de $\mathcal{P}_r$ par ψ est

une fonction de degré au plus r, il suffit en vertu de l'hypothèse de récurrence et de la proposition 19 ci-dessous de montrer que pour tout $u \in F_{2^m}$, $P(X) + P(X+u)$ appartient à $\mathcal{P}_{r-1}$, ce qui se vérifie facilement. $\diamond$

Proposition 19 *Une fonction booléenne sur F_2^m appartient au code RM(r,m) si et seulement si, pour tout mot u, la fonction $f(x) + f(x+u)$ appartient au code RM(r − 1,m).*
Cette propriété s'étend aux fonctions vectorielles.

Preuve: Il est facile de voir que si f est de degré au plus r, alors $f(x)+f(x+u)$ est de degré au plus $r-1$: il suffit de le montrer pour les monômes, ce qui ne pose pas de problème.
Montrons maintenant la réciproque. Supposons que f soit de degré $d \geq r+1$. On peut, sans perte de généralité supposer que la forme algébrique normale de f contient le monôme $\prod_{i=1}^{d} x_i$. Soit alors $u = (1,0,\ldots,0)$. La forme algébrique normale de $f(x) + f(x+u)$ contient le monôme $\prod_{i=2}^{d} x_i$ et a pour degré $d-1 \geq r$. Cela termine la démonstration. $\diamond$

Preuve du théorème:
Soit λ un élément de F_{2^m} tel que $tr(\lambda) = 1$.
Toute fonction booléenne sur F_{2^m} est une fonction vectorielle à valeurs dans F_{2^m} (puisque F_{2^m} contient F). Il suffit alors de remarquer que:

$$f(x) = tr(\lambda f(x))$$

pour conclure à l'existence d'une telle représentation des fonctions de RM(r,m). La réciproque est à peu près évidente. $\diamond$

Le code de Kerdock est non linéaire car on peut montrer que, si u et v sont linéairement indépendants, la fonction $f_u + f_v$ n'appartient pas à $\mathcal{K}_m$. Notons que les fonctions de RM(1,m) étant les fonctions de la forme:

$$h(x,\epsilon) = tr(ax) + \epsilon\eta + \mu$$

où $a \in F_{2^{m-1}}$ et $\eta,\mu \in F_2$, les éléments de $\mathcal{K}_m$ sont les fonctions de la forme:

$$f_u(x,\epsilon) = tr\left(\sum_{i=1}^{t} (ux)^{2^i+1}\right) + tr(ax) + \epsilon(tr(ux) + \eta) + \mu.$$

Proposition 20 *La distance minimale de $\mathcal{K}_m$ est $2^{m-1} - 2^{\frac{m}{2}-1}$.*

Preuve: Montrer que la distance minimale de $\mathcal{K}_m$ est égale au rayon de recouvrement de RM(1,m) revient à montrer que toute fonction non affine de

ce code est courbe et que la somme, supposée non affine, de deux fonctions de ce code est courbe également.

Lemme 3 *Soit f une fonction booléenne quelconque sur F_2^m, alors la somme (dite* somme de caractères*) suivante:*

$$S_f = \sum_{x \in F_2^m} (-1)^{f(x)}$$

est égale à $2^m - 2\,\omega(f)$, où $\omega(f)$ désigne le poids de f.
La preuve, évidente, est laissée au lecteur.
Remarquons que si f est une forme linéaire sur F_2^m, alors S_f est égale à 2^m si $f = 0$ et à 0 sinon. Plus généralement:
Exercice 43 *Soit E un sous-espace vectoriel de F_2^m et f une forme linéaire non nulle sur E, montrer que la somme $\sum_{x \in E}(-1)^{f(x)}$ est nulle.*
Notons que, pour toute fonction booléenne f, S_{f+1} est égale à $-S_f$.

Soit maintenant f une fonction quadratique, dont on veut calculer le poids. D'après le lemme ci-dessus, il suffit de déterminer S_f. Il va être plus simple de calculer ${S_f}^2$. Cette perte d'information sur le signe de S_f n'entraine pas de perte d'information sur la distribution des poids des fonctions quadratiques puisque, leur ensemble étant stable par addition de la fonction constante 1, la distribution des valeurs de S_f est stable par multiplication par -1.
Proposition 21 *Soit f une fonction quadratique. Alors le nombre ${S_f}^2$ est égal à:*
. $2^{m+\dim \mathcal{E}_f}$ *si la restriction de f à $\mathcal{E}_f$ est constante,*
. 0 *sinon.*
Preuve: On a :

$$ {S_f}^2 = \sum_{(x,y) \in F_2^{m\,2}} (-1)^{f(x)+f(y)}. $$

On ne change pas la valeur de cette somme en remplaçant x par $x+y$, puisque l'application $(x,y) \to (x+y,y)$ est une bijection de $F_2^{m\,2}$ sur lui-même.
On a donc, en notant $\varphi_f(x,y) = f(x+y) + f(y) + f(x) + f(0)$ la *forme symplectique* associée à f:

$$ {S_f}^2 = \sum_{(x,y) \in F_2^{m\,2}} (-1)^{f(x+y)+f(y)} = \sum_{(x,y) \in F_2^{m\,2}} (-1)^{\varphi_f(x,y)+f(x)+f(0)} = $$

$$\sum_{x\in F_2^m}\left((-1)^{f(x)+f(0)}\sum_{y\in F_2^m}(-1)^{\varphi_f(x,y)}\right),$$

où $\varphi_f(x,y) = f(x+y)+f(y)+f(x)+f(0)$ est bilinéaire (exercice!). Pour chaque x, la somme $\sum_{y\in F_2^m}(-1)^{\varphi_f(x,y)}$ est la somme de caractères associée à la forme linéaire $y \to \varphi_f(x,y)$. Elle est donc non nulle si et seulement si x appartient au noyau $\mathcal{E}_f = \{y \in F_2^m / \forall x \in F_2^m, \varphi_f(x,y) = 0\}$, auquel cas elle vaut 2^m. On en déduit:

$$S_f{}^2 = 2^m \sum_{x\in\mathcal{E}_f}(-1)^{f(x)+f(0)}.$$

Or, la restriction de la fonction $x \to f(x)+f(0)$ au noyau $\mathcal{E}_f$ est linéaire. D'après l'exercice 43, la somme $\sum_{x\in\mathcal{E}_f}(-1)^{f(x)+f(0)}$ est donc nulle si et seulement si cette restriction est non identiquement nulle, c'est à dire si et seulement si la restriction de f à $\mathcal{E}_f$ est non constante. Dans le cas contraire, cette somme vaut $|\mathcal{E}_f|$. Cela termine la démonstration. ◇

Calculons la forme symplectique associée à une fonction quelconque de $\mathcal{K}_m$.

Lemme 4 *La forme symplectique associée à la fonction:*

$$f_u(x,\epsilon) = tr\left(\sum_{i=1}^{t}(ux)^{2^i+1}\right)+\epsilon tr(ux)$$

est:

$$\varphi_{f_u}((x,\epsilon),(y,\eta)) = tr(ux)tr(uy)+tr(u^2xy)+\epsilon tr(uy)+\eta tr(ux).$$

Preuve:

$$\varphi_{f_u}((x,\epsilon),(y,\eta)) = f_u(0,0)+f_u(x,\epsilon)+f_u(y,\eta)+f_u(x+y,\epsilon+\eta) =$$

$$tr\left(\sum_{i=1}^{t}(ux)^{2^i+1}\right)+\epsilon tr(ux)+tr\left(\sum_{i=1}^{t}(uy)^{2^i+1}\right)+\eta tr(uy)+$$

$$tr\left(\sum_{i=1}^{t}(u(x+y))^{2^i+1}\right)+(\epsilon+\eta)tr(u(x+y)).$$

On a:

$$(u(x+y))^{2^i+1} = u^{2^i+1}\left(x^{2^i+1} + y^{2^i+1} + x^{2^i}y + y^{2^i}x\right).$$

D'où:

$$tr\left(\sum_{i=1}^{t}(ux)^{2^i+1}\right) + tr\left(\sum_{i=1}^{t}(uy)^{2^i+1}\right) + tr\left(\sum_{i=1}^{t}(u(x+y))^{2^i+1}\right) =$$

$$tr\left(\sum_{i=1}^{t}u^{2^i+1}\,x^{2^i}y\right) + tr\left(\sum_{i=1}^{t}u^{2^i+1}\,y^{2^i}x\right).$$

Rappelons que pour tout élément v de $F_{2^{m-1}}$, et pour tout entier k, on a $tr(v) = tr(v^{2^k})$. Appliquons cette propriété avec $k = m' - i, m' = m - 1$, on obtient:

$$tr\left(\sum_{i=1}^{t}u^{2^i+1}\,x^{2^i}y\right) + tr\left(\sum_{i=1}^{t}u^{2^i+1}\,y^{2^i}x\right) =$$

$$tr\left(\sum_{i=1}^{t}u^{2^i+1}\,x^{2^i}y\right) + tr\left(\sum_{i=1}^{t}u^{2^{m'-i}+1}\,x^{2^{m'-i}}y\right) =$$

$$tr\left(\sum_{i=1}^{t}\left((ux)^{2^i}(uy)\right)\right) + tr\left(\sum_{i=t+1}^{m'-1}\left((ux)^{2^i}(uy)\right)\right) =$$

$$tr\left(\sum_{i=1}^{2t}\left((ux)^{2^i}(uy)\right)\right) =$$

$$tr\left[[ux + tr(ux)](uy)\right] = tr(ux)tr(uy) + tr(u^2xy).$$

Le résultat s'en déduit facilement. ◇

Fin de la preuve de la proposition 20:
Il nous faut d'abord calculer le noyau $\mathcal{E}_{f_u}$ de φ_{f_u} et montrer que, si $u \neq 0$, il est réduit à $\{0\}$.
On a:

$$\mathcal{E}_{f_u} = \{(x,\epsilon) \in F_{2^{m-1}} \times F_2) \,|\, \forall (y,\eta) \in F_{2^{m-1}} \times F_2), \varphi_{f_u}((x,\epsilon),(y,\eta)) = 0\}$$

$$= \{(x,\epsilon) \,|\, \forall (y,\eta), tr(ux)tr(uy) + tr(u^2xy) + \epsilon tr(uy) + \eta tr(ux) = 0\}$$

$$= \{(x,\epsilon) \,|\, \forall (y,\eta), tr\left([utr(ux) + u^2x + \epsilon u]y\right) = 0 \text{ et } \eta\, tr(ux) = 0\}.$$

On sait que pour tout v, la fonction $tr(vy)$ est identiquement nulle si et seulement si $v = 0$. Donc (x,ϵ) appartient à $\mathcal{E}_{f_u}$ si et seulement si:

$$utr(ux) + u^2x + \epsilon u = 0$$

$$\text{et } tr(ux) = 0$$

soit encore si:

$$ux + \epsilon = 0 \tag{9.1}$$

$$\text{et } tr(ux) = 0. \tag{9.2}$$

En appliquant la fonction trace à l'égalité 9.1, on déduit $\epsilon = 0$, m' étant impair, ce qui implique alors $x = 0$.

Calculons maintenant le noyau $\mathcal{E}_{f_u+f_v}$ de $\varphi_{f_u+f_v}$ et montrons que, si $u \neq v$, il est réduit à $\{0\}$.
De même que précédemment, (x,ϵ) appartient à $\mathcal{E}_{f_u+f_v}$ si et seulement si:

$$utr(ux) + vtr(vx) + (u^2 + v^2)x + \epsilon(u + v) = 0$$

$$\text{et } tr((u + v)x) = 0$$

soit encore si:

$$(u + v)tr(ux) + (u^2 + v^2)x + \epsilon(u + v) = 0$$

$$\text{et } tr((u + v)x) = 0$$

soit enfin si:

$$tr(ux) + (u + v)x + \epsilon = 0 \tag{9.3}$$

$$\text{et } tr((u + v)x) = 0. \tag{9.4}$$

En appliquant la fonction trace à la relation 9.3, on obtient $tr(ux) + \epsilon = 0$ qui, d'après 9.3, implique $x = 0$ puis $\epsilon = 0$. $\diamond$

La distribution de poids du code $\mathcal{K}_m$ n'est maintenant pas difficile à obtenir.

Exercice 44 *Montrer que le code $\mathcal{K}_m$ est distance-invariant.*
Montrer que pour ce code, on a: $A_0 = 1$, $A_{2^{m-1}-2^{\frac{m}{2}-1}} = A_{2^{m-1}+2^{\frac{m}{2}-1}} = 2^m(2^{m-1}-1)$, $A_{2^{m-1}} = 2^{m+1}-2$, et $A_{2^m} = 1$, et que tous les autres coefficients A_i sont nuls.

Il a été montré, dès 1969, par Semakov et Zinoviev, que la transformation de Mac Williams transforme le polynôme énumérateur des poids du code $\mathcal{K}_m$ en celui d'un code appelé le code de Preparata (qu'on n'introduira pas ici). Une telle *dualité formelle* entre des classes de codes non linéaires a beaucoup intrigué les chercheurs depuis cette date. La question était de savoir s'il

existait entre ces codes une dualité algébrique (à définir puisqu'ils sont non linéaires).
Certains, comme William Kantor, n'hésitaient pas à affirmer qu'il s' agissait d'une coïncidence.
En 1994 est paru un article[1], qui montre que le code de Kerdock est l'image par une certaine transformation isométrique (en un sens que l'on précisera plus loin) d'un code linéaire sur l'anneau $\mathbf{Z}_4$ des entiers modulo 4 (i.e. d'un ensemble stable par addition dans $\mathbf{Z}_4$). On appelle $\mathbf{Z}_4$-linéaires de tels codes. Ils sont en général non linéaires, en tant que codes binaires, mais admettent des duaux (en un sens qu'on va préciser). Depuis, on a généralisé cette construction à $\mathbf{Z}_{2^k}$ pour tout k positif[2].

9.2 Codes $\mathbf{Z}_4$-linéaires

Définition 32 *On appelle code quaternaire tout code linéaire sur* $\mathbf{Z}_4$.

Remarque 18 *On se méfiera de cette terminologie, car on trouve aussi dans la littérature le terme de code quaternaire pour désigner les codes linéaires sur* F_4.

Un code quaternaire de longueur n est donc un sous-ensemble de ${\mathbf{Z}_4}^n$ qui est stable par addition, et donc également stable par multiplication externe par les éléments de $\mathbf{Z}_4$.
La notion de matrice génératrice existe encore dans le cadre des codes quaternaires, bien que ces codes ne soient pas toujours des modules libres. Les lignes d'une matrice génératrice engendrent par combinaisons linéaires les mots du code, mais il n'y a pas unicité de la décomposition.
On montre que tout code quaternaire est l'image, par une certaine transformation affine, d'un code de matrice génératrice:

$$\begin{bmatrix} I_{k_1} & A & B \\ 0 & 2I_{k_2} & 2C \end{bmatrix}$$

1. A. R. Hammons Jr., P. V. Kumar, A. R. Calderbank, N. J. A. Sloane et P. Solé, *The Z_4-linearity of Kerdock, Preparata, Goethals and related codes* IEEE Transactions on Information Theory, vol 40, pp 301-320, (1994)

2. C. Carlet, *Z_{2^k}-linear codes* IEEE Transactions on Information Theory, Vol. 44, n° 4, pp. 1543-1547 (1998)

où A et C sont des matrices sur $\mathbf{Z}_2$ et où B est une matrice sur $\mathbf{Z}_4$.
On définit le produit scalaire usuel sur ${\mathbf{Z}_4}^n$:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i \mod 4$$

et la notion de code dual d'un code quaternaire qui en découle:

$$C^{\perp} = \{b \in {\mathbf{Z}_4}^n \mid \forall a \in C, a \cdot b = 0\}.$$

9.2.1 Polynômes énumérateurs des poids des codes quaternaires

On définit, comme pour les codes sur les corps finis, le polynôme énumérateur des poids complets d'un code quaternaire C de longueur n:

$$CW_C(X_0,X_1,X_2,X_3) = \sum_{a \in C} \prod_{i=1}^{n} X_{a_i},$$

et le polynôme énumérateur des poids de Hamming:

$$HW_C(X,Y) = \sum_{a \in C} X^{n-\omega(a)} Y^{\omega(a)},$$

où $\omega(a)$ désigne le poids de Hamming du mot a. HW_C peut s'obtenir à partir du polynôme énumérateur des poids complets en remplaçant X_0 par X et X_1, X_2 et X_3 par Y.
On définit aussi le polynôme énumérateur des poids de Lee:

$$LW_C(X,Y) = \sum_{a \in C} X^{2n-\omega_L(a)} Y^{\omega_L(a)},$$

où $\omega_L(a)$ désigne le poids de Lee du mot a. Ce poids est égal à la somme $\sum_{i=1}^{n} j_i$ dans laquelle j_i est égal à 0 si $a_i = 0$, à 1 si $a_i = 1$ ou 3, et à 2 si $a_i = 2$.
LW_C va être en fait le polynôme énumérateur de Hamming de l'image binaire du code C. Il peut s'obtenir à partir du polynôme énumérateur des poids complets en remplaçant X_0 par X^2, X_1 et X_3 par XY, et X_2 par Y^2.
Exercice 45 *Montrer que l'on a:*

$$CW_C(X_0+X_1+X_2+X_3,X_0+iX_1-X_2-iX_3,X_0-X_1+X_2-X_3,X_0-iX_1-X_2+iX_3) =$$

$$\sum_{a\in C}\prod_{i=1}^{n}\left(\sum_{j=0}^{3} i^{ja_i}X_j\right) = \sum_{a\in C, b\in \mathbf{Z}_4^n}\prod_{i=1}^{n}\left(i^{b_i a_i}X_{b_i}\right).$$

En déduire:

1. $CW_{C^\perp} = \frac{1}{|C|}CW_C(X_0+X_1+X_2+X_3,$

   $X_0+iX_1-X_2-iX_3, X_0-X_1+X_2-X_3, X_0-iX_1-X_2+iX_3),$

2. $HW_{C^\perp} = \frac{1}{|C|}HW_C(X+3Y, X-Y)$ *et:*
3. $LW_{C^\perp} = \frac{1}{|C|}LW_C(X+Y, X-Y).$

9.2.2 Le "Gray map"

On va définir maintenant la transformation isométrique qui, à un code quaternaire de longueur n, fait correspondre un code binaire de longueur $2n$. On appellera ce code *l'image binaire* du code quaternaire. Cette transformation s'appelle le "Gray map", car elle est liée au code de Gray (i.e. l'énumération des mots binaires telle que, en passant d'un mot à son successeur, on n'ait à changer qu'un seul bit).

Définition 33 *On appelle "Gray map" l'application de* $\mathbf{Z}_4$ *dans* ${\mathbf{Z}_2}^2$ *qui à* 0 *fait correspondre* (0,0)*, à* 1 *fait correspondre* (0,1)*, à* 2 *fait correspondre* (1,1)*, et à* 3 *fait correspondre* (1,0)*. On appelle plus généralement "Gray map" l'extension de cette application aux mots sur* $\mathbf{Z}_4$*.*

En d'autres termes, le "Gray map" est l'application:

$$2a+b \rightarrow (a, a\oplus b)$$

où "$\oplus$" est l'addition modulo 2, et où $2a+b$ est l'écriture (unique) d'un élément de $\mathbf{Z}_4$ (ou plus généralement de ${\mathbf{Z}_4}^n$) en fonction de deux éléments de $\mathbf{Z}_2$ (de ${\mathbf{Z}_2}^n$).

L'inverse du "Gray map" (qu'on appellera *relèvement*) est l'application:

$$(u,v) \rightarrow 2u + (u\oplus v)$$

où "+" est l'addition dans $\mathbf{Z}_4$.

Exercice 46 *Vérifier que le "Gray map" est une isométrie de* ${\mathbf{Z}_4}^n$*, muni de la distance de Lee, sur* ${\mathbf{Z}_2}^{2n}$*, muni de la distance de Hamming.*

Définition 34 *On appelle code $\mathbf{Z}_4$-linéaire tout code équivalent, modulo une permutation des coordonnées, à l'image par le "Gray map" d'un code quaternaire. Le code quaternaire ainsi défini s'appelle un relevé du code.*
On appelle $\mathbf{Z}_4$-dual d'un tel code C l'image $C_\perp$ par le "Gray map" du dual d'un relevé de C.

Exercice 47 *Montrer que tout code $\mathbf{Z}_4$-linéaire est distance-invariant. Montrer que le polynôme énumérateur des poids d'un tel code est l'image par la transformation de Mac Williams de celui de son $\mathbf{Z}_4$-dual.*

9.2.3 La $\mathbf{Z}_4$-linéarité des codes de Kerdock

Pour éviter les confusions, on va encore noter $\oplus$ l'addition dans $\mathbf{Z}_2$ pour la distinguer de l'addition dans $\mathbf{Z}_4$.

Proposition 22 *Un code de longueur $2n$ est $\mathbf{Z}_4$-linéaire si et seulement si il est équivalent, modulo une permutation des coordonnées, à un code C tel que:*

$$\forall\, u_1 \,|\, v_1 \in C, \forall\, u_2 \,|\, v_2 \in C,$$

$$u_1 \,\oplus\, u_2 \,\oplus\, (u_1 \oplus v_1)(u_2 \oplus v_2) \,|\, v_1 \,\oplus\, v_2 \,\oplus\, (u_1 \oplus v_1)(u_2 \oplus v_2) \in C.$$

Preuve: Notons $\hat{C}$ le relevé de C. Le relevé d'un mot $u \,|\, v$ du code C est le mot sur $\mathbf{Z}_4$ égal à $2u+(u \oplus v)$. Donc $\hat{C}$ est un code quaternaire si et seulement si:

$$\forall\, u_1 \,|\, v_1 \in C,\, \forall\, u_2 \,|\, v_2 \in C,\, 2u_1 + (u_1 \oplus v_1) + 2u_2 + (u_2 \oplus v_2) \in \hat{C}$$

soit encore (puisque $2(u_1 + v_1) = 2(u_1 \oplus v_1)$):

$$2(u_1 \oplus u_2) + (u_1 \oplus v_1 \oplus u_2 \oplus v_2) + 2(u_1 \oplus v_1)(u_2 \oplus v_2) \in \hat{C}.$$

$\diamond$

Nous reprenons maintenant l'écriture habituelle de l'addition dans F_2^m. Considérons deux éléments quelconques du code de Kerdock:

$$tr\left(\sum_{i=1}^{t}(u_1 x)^{2^i+1}\right) + tr(a_1 x) + \epsilon(tr(u_1 x) + \eta_1) + \mu_1$$

et

$$tr\left(\sum_{i=1}^{t}(u_2 x)^{2^i+1}\right) + tr(a_2 x) + \epsilon(tr(u_2 x) + \eta_2) + \mu_2.$$

D'après la proposition précédente, pour montrer la $\mathbf{Z}_4$-linéarité du code de Kerdock, il suffit de vérifier que la fonction:

$$tr\left(\sum_{i=1}^{t}(u_1x)^{2^i+1}\right)+tr(a_1x)+\epsilon(tr(u_1x)+\eta_1)+\mu_1+$$

$$tr\left(\sum_{i=1}^{t}(u_2x)^{2^i+1}\right)+tr(a_2x)+\epsilon(tr(u_2x)+\eta_2)+\mu_2+(tr(u_1x)+\eta_1)(tr(u_2x)+\eta_2)$$

appartient à $\mathcal{K}_m$.

Exercice 48 *Vérifier cette propriété.*

On a donc une explication de la dualité formelle de $\mathcal{K}_m$ avec un autre code (le code de Preparata). Mais on a pu vérifier que le $\mathbf{Z}_4$-dual de $\mathcal{K}_m$ n'est pas équivalent à $\mathcal{P}_m$, modulo un automorphisme affine de F_2^m. Il reste donc encore une petite partie du voile à lever quant à la question d'une dualité algébrique entre $\mathcal{K}_m$ et $\mathcal{P}_m$.

9.2.4 La représentation du code de Kerdock par la fonction trace de l'anneau de Galois

Il existe une généralisation à l'anneau $\mathbf{Z}_4$ (et plus généralement à tout anneau $\mathbf{Z}_{p^r}$) de la technique d'extension (on dit aussi "de relèvement") des corps premiers. On définit ainsi les *anneaux* de Galois, qui constituent une sur-classe de la classe des corps de Galois (i.e. des corps finis) et de celle des anneaux $\mathbf{Z}_{p^r}$, et qui possèdent essentiellement les mêmes propriétés que les corps finis (voir le problème 1 du TD 3):

Soit $h(X)$ un polynôme primitif de degré m sur F_2 (admettant pour racine un élément primitif α du corps F_{2^m}). Soit $n=2^m-1$. On rappelle que $h(X)$ divise le polynôme X^n-1 dans $F_2[X]$. En posant $h(X)=e(X)+d(X)$ où le polynôme $e(X)$ (resp. $d(X)$) a tous ses exposants pairs (resp. impairs), on définit $k(X^2)=d^2(X)-e^2(X)$ où les opérations de soustraction et de mise aux carré sont effectuées dans $\mathbf{Z}_4$. Alors $k(X)$ divise X^n-1 dans $\mathbf{Z}_4[X]$. On appelle anneau de Galois $GR(4^m)$ l'algèbre quotient $\mathbf{Z}_4[X]/(k(X))$. On désigne par ξ le représentant du polynôme X dans l'algèbre $\mathbf{Z}_4[X]/(k(X))$ (racine dite primitive de $k(X)$). Chaque élément de $GR(4^m)$ s'écrit de façon unique sous la forme $\sum_{i=0}^{m-1}\lambda_i\xi^i$ ($\lambda_i\in\mathbf{Z}_4$) et sous la forme $a+2b$ où a et b appartiennent à l'ensemble $\mathcal{T}=\{0,1,\xi,\xi^2,\ldots,\xi^{n-1}\}$, égal à l'ensemble des carrés des éléments de $GR(4^m)$.

Remarque: On peut identifier chaque élément ξ^i de $\mathcal{T}$ à l'élément α^i de F_{2^n} et on a alors $GR(4^m) = F_{2^m} + 2F_{2^m}$ (attention, l'opération d'addition dans $GR(4^m)$ correspond à celle d'addition dans F_{2^m} pour ce qui est de b mais ne correspond pas à celle d'addition dans F_{2^m} pour ce qui est de a).

L'application:

$$\phi : a + 2b \rightarrow a^2 + 2b^2,$$

où a et b décrivent $\mathcal{T}$, est un automorphisme d'anneaux, encore appelé *automorphisme de Frobenius* et on peut, comme dans le cas des corps finis, définir une fonction trace:

$$Tr(\beta) = \beta + \phi(\beta) + \phi^2(\beta) + \ldots + \phi^{m-1}(\beta)$$

où $\phi^{m-1} = \phi \circ \ldots \circ \phi$.

Exercice 49 *Montrer que, pour tout $\beta \in GR(4^m)$, on a $Tr(\beta) \in \mathbf{Z}_4$.*

Exercice 50 *1. Soit i un entier compris entre 0 et $n-1$. Soit a_0,$b_0 \in \{0,1\}$ et a,$b \in \mathcal{T}$. Calculer à l'aide de la fonction trace dans F_{2^m} (qu'on notera tr) les valeurs de c et d (considérés comme éléments de F_{2^m}) tels que $Tr((a+2b)\xi^i) + a_0 + 2b_0 = c + 2d$.*
Indication*: calculer le carré de $Tr((a+2b)\xi^i) + a_0 + 2b_0$, et en déduire c, puis calculer $Tr((a+2b)\xi^i) + a_0 + 2b_0 - c$ et en déduire b.*
2. Montrer que l'image par le Gray map du code linéaire sur $\mathbf{Z}_4$ égal à l'ensemble des mots de la forme

$$(u, Tr(\beta) + u, Tr(\beta\xi) + u, Tr(\beta\xi^2) + u, \ldots, Tr(\beta\xi^{n-1} + u))$$

est égal au code de Kerdock.