SOMMAIRE

INTRODUCTION

Depuis le commencement les hommes ont été confrontés au défi majeur de la communication. De l'Ouest à l'Est, du Nord au Sud, tous visent à faire reposer leur processus de communication sur les piliers Confidentialité, Authenticité, Intégrité et la Non répudiation en abrégé (CAIN). Dans le cadre de notre analyse, nous nous appesantirons exclusivement sur l'intégrité du message transmis. En effet, lors de la transmission d'un message à travers un canal peu fiable, des erreurs peuvent s'y introduire et corrompe le message transmis. Dès-lors, deux interrogations suscitent notre attention. Peut-on retrouver à partir du message erroné ou corrompu, le message initial ? Si oui, Comment récupérer l'information ? Répondre à ces questions à l'aide des algorithmes de décodage par ensemble d'information est le but de notre devoir.

I. PRELIMINAIRE : FORMALISME MATHEMATIQUE

Les mathématiques sont partout dans l'encodage de l'information, laquelle se fait en transformant les informations en suites de bits, soit des 0 et des 1. Mais, lors du transfert de l'information, certains bits peuvent être corrompus, un 0 étant transformé en 1, ou le contraire. Le décodage par ensemble d'information qui est une technique permettant de retrouvant l'information perdue, repose sur la théorie des codes linéaires, qui témoigne de la puissance de l'algèbre linéaire. Pour pouvoir donc aborder plus aisément notre analyse, il est impératif de rappeler de façon succincte, de définir quelques concepts mathématiques et de préciser les notations qui seront utilisées par la suite.

A. Rappels d'éléments mathématiques

1. La théorie des corps finis

Un corps commutatif est un ensemble dans lequel les 4 opérations d'addition, multiplication, soustraction et division fonctionnent comme nous en avons l'habitude quand nous utilisons les nombres réels (R) ou complexes (C). Plus précisément :

Définition : Soit $(K, +, \cdot)$ un ensemble muni de deux opérations binaires notées $+$ et $\cdot$. Nous disons que c'est un corps commutatif (Anglais. Field) si :

1. L'addition fait de K un groupe commutatif. Son élément neutre est noté 0.
2. La multiplication fait de l'ensemble K privé de 0 un groupe commutatif. En particulier, tous les éléments sauf 0 sont inversibles. L'élément neutre de la multiplication est noté 1.
3. La multiplication est distributive par rapport à l'addition : $a \cdot (x + y) = a \cdot x + a \cdot y$ pour tous $a, x, y \in K$.

Les ensembles R et C sont des corps commutatifs infinis. Nous nous intéressons aux corps commutatifs finis, qui peuvent être utilisés comme alphabet par un code correcteur ou détecteur. Nous notons $F_2 = (\{0,1\}, +, \cdot)$ le corps fini à deux éléments. Sur ce corps, ces opérations sont modulos 2.

Théorème : soit Z/pZ si p est un nombre premier, tous les éléments de Z/pZ sauf $[0]_p$ sont inversibles. Il est facile de voir que cela entraîne que $(Z/pZ, +, \cdot)$ est un corps commutatif.

Théorème et Définition Dans un corps fini, il existe un plus petit entier $p > 0$ tel que $p \cdot 1 = 0$, c'est à dire tel que p fois $1_K + 1 + ... + 1_K = 0$. Ce nombre est un nombre premier. Il est appelé la caractéristique du corps.

2. Espace vectoriel

Définition : Soit K un corps commutatif et (V, +) un groupe commutatif, muni d'une opération binaire notée +. Supposons qu'une opération externe est définie sur K et V, c'est à dire une application qui à $\lambda \in K$ et $x \in V$ associe un élément, noté λx de V. Les éléments du corps commutatif sont appelés scalaires et les éléments de V sont appelés vecteurs. L'opération externe est appelée multiplication scalaire. Nous disons que V muni de ces deux opérations est un espace vectoriel sur le corps K si les propriétés suivantes sont vraies : pour tous scalaires λ, μ et vecteurs u,v :

- Associativité pour la multiplication scalaire : $\lambda(\mu.v) = (\lambda\mu)v$
- Identité : $1 \cdot v = v$
- Distributivité : $\lambda(u + v) = \lambda u + \lambda v$ et $(a + b)u = au + bu$

Dans notre analyse, nous allons utiliser les espaces vectoriels $V = K^n$ où K est un corps fini. Un sous-ensemble S de V est un sous-espace vectoriel, s'il est aussi un espace vectoriel sur K.

Définition(base) : Une base est une famille de vecteurs libres et génératrice de l'espace vectoriel.

Définition (dimension) : Si $V = K^n$ ou si V est un sous-espace vectoriel de K^n, il possède des bases finies, et toutes les bases ont le même cardinal, appelé la **dimension** de l'espace vectoriel, notée **dim(V).** Par convention, la dimension de l'espace vectoriel nul c'est zéro.

3. Alphabet

Afin de préciser les questions que se pose la théorie des codes, et les problèmes qu'elle rencontre, notre étude considère le cas d'un canal discret. L'information à transmettre peut être vue comme une suite *x* de symboles pris dans un corps(ensemble) fini (il s'agit le plus souvent de bits, donc de 0 et de 1).

-**Un alphabet** est un ensemble fini non vide, ses éléments sont appelés lettres ou symboles. C'est un ensemble fini de symboles avec lequel sont écrits les mots du code. On prendra pour alphabet F_q , un corps fini à q éléments muni des opérations somme et produit.

-**Un message A** ou un mot est une suite à valeur dans un alphabet, il correspond à une suite de lettres. A est une liste de k symboles à transmettre, on l'assimile à un élément de F_q^k.

Remarque : q dans notre analyse vaut 2. A est donc l'alphabet binaire.

B. THEORIE DES CODES

1. Définitions générales

Caractéristiques d'un code : afin de formaliser la notion de code correcteur, nous utiliserons les définitions suivantes :

- **L'encodeur** est un algorithme qui transforme le message *A* en un mot du code *C* de longueur *n* comportant des redondances. On l'assimile à une fonction injective f : $F_q^k \rightarrow F_q^n$

-**Le code de paramètre** *(k, n)* est l'ensemble des mots possibles : C={ f(B), B dans F_q^k }. C'est un sous ensemble de F_q^n. Si C est un sous-espace vectoriel de F_q^n C est appelé **Code linéaire**. K c'est la dimension du code et n sa longueur. **C** est appelé image de f.

2. Distance de Hamming

Définition : Soit A un ensemble fini (l'alphabet) et n ≥ 1 un entier. Soient $x = (x_1, ..., x_n) \in A^n$ et $y = (y_1, ..., y_n) \in A^n$ deux suites de n éléments de A. La distance de Hamming d(x, y) est le nombre de positions où x et y diffèrent :

d(x, y) = card {i ∈ {1, ..., n} tels que $x \neq y$ }

x = (1, 0, 1, 1, 1, 0) et y = (1, 0, 0, 1, 1, 1) ne diffèrent qu'en leur troisième et dernière positions, donc leur distance de Hamming est d(x, y) = 2. Notons que le cardinal de l'ensemble vide est 0 donc d(x, x) = 0.

Exemple : la distance de Hamming entre 1011101 et 1001001 est 2 car deux bits sont différents.

Théorème : (Distance de Hamming) La distance de Hamming est une distance sur A^n.

Preuve :

1-Par définition, d(x,y)≥0 et d(x,y)=0 alors ∀i ∈{1 ,2,…,n} $x_i=y_i \Rightarrow x=y$.

2- d(x,y)= card {i ∈ {1, ..., n} tels que $x \neq y$ }=d(y,x)

3- Soit A l'ensemble des positions où x et y diffèrent, B l'ensemble des positions où y et z diffèrent et C l'ensemble des positions où x et z diffèrent.

Nous avons donc d (x, y) = card(A), d(y, z) = card(B) et d(x, z) = card(C). Or :($x_i = y_i$ et $y_i = z_i$) ⇒ ($x_i = z_i$) donc par contraposition

($x_i \neq z_i$) ⇒ ($x_i \neq y_i$ ou $y_i \neq z_i$) c'est à dire que ($i \in C$) ⇒ ($i \in A$) ou ($i \in B$) ou encore

$C \subset (A \cup B)$ donc $card(C) \leq card(A \cup B) \leq card(A) + card(B)$. Ce qu'il fallait démontrer.

Remarque préliminaire :

Dans la suite, le mot distance désignera la distance de Hamming.

Il y a 2^k mots de k bits.

Le principe du codage est le suivant : après avoir découpé notre message en blocs de *k* bits, on va appliquer un même algorithme sur chaque bloc :

a) Ou bien en rajoutant des bits de contrôle `à la fin de chaque bloc

b) Ou bien en modifiant complètement les blocs, mais en évitant que deux blocs différents soient transformés en un même bloc. D'où la définition suivante :

Définition (poids de Hamming) : On appelle **poids de m**, et on note **w(n)** le nombre de lettres non nulles de m (donc le nombre de bits de m égaux à 1).

Anecdotiquement, on a les deux relations suivantes : d (m, m′) = w (m + m′) et donc w(m) = d (m, 00...0), ou` + désigne l'addition bit à bit (avec 1 + 1 = 0 . . .).

Remarque : Le poids d'une erreur est le nombre de positions qui sont modifiées.

Définition : Un code vérifie la **condition de décodage d'ordre e** si pour tout x de , il existe au plus un mot y de F_q^n , d(x,y)≤e. Cette condition est équivalente à toute les boules de fermées pour la distance sont deux à deux disjointes.

3. Distance minimale et rayon de recouvrement

Soit f un encodeur d'image C.

On appelle **capacité de détection** de f le plus grand entier e_d tel qu'on soit toujours capable de détecter e_d erreurs ou moins.

On appelle capacité de correction de φ le plus grand entier e_c tel qu'on soit toujours capable de corriger e_c erreurs ou moins.

On appelle **distance minimale de C** et on note **d** la plus petite distance non nulle entre deux mots de code. C'est égal au poids minimal.

$$\mathbf{d = min\{d (x , y) : x , y \in C , x \neq y\}} = \min\{w(c)\}$$

On appelle **rayon de recouvrement de C** la quantité t=$\left\lfloor \frac{d-1}{2} \right\rfloor$. C'est la capacité maximale de correction.

Notation :

La distance minimale d'un code quantifie donc sa qualité vis à vis du point **1**. C'est un paramètre important. En abrégé, un code de dimension *k*, de longueur *n* et de distance minimale *d* se dira "un code de paramètres (*k, n, d*) ou même un (*k, n, d*) –code ou (n,k,d)-code selon le formalisme employé. Dans la suite, nous utiliserons sauf en cas d'indication contraire, la notation (n,k,d)-code.

Propositions :

(1)-On dit qu'un code C détecte jusqu'à e erreurs si et seulement si pour tout **x, y ∈ C, x ≠y alors d(x,y)≥e+1 .** Autrement dit, un code détecte des erreurs **si et seulement si toutes les erreurs sont de poids p < dmin(C).**

(2)-On dit qu'un code C corrige jusqu'à e erreurs si et seulement si pour tout **x, y ∈ C, x ≠y alors d(x,y)≥2e+1 .** Autrement dit, un code corrige des erreurs **si et seulement si toutes les erreurs sont de poids p <** $\frac{d_{min}(C)}{2}$**.**

(3)-Soit C code de rayon t et y un élément de F_q^n, Il existe un unique c de C tel que d(y,c)≤t.

(4) - (borne de Singleton) : La distance minimale d'un code linéaire de dimension k et de longueur n vérifie d ≤ n + 1 − k. Un code pour lequel on a égalité est dit MDS (Maximum Distance Séparable).

(5) - (Inégalité de Hamming) : Soit f un encodeur de paramètres (k, n), d'image C, de capacité de correction e_c. $\sum_{i=0}^{ec} \binom{n}{i} \leq 2^{n-k}$

Preuve (propositions 1, 2, 4) :

(1) - Soit x un mot de code transmis et y le mot reçu, avec erreur de poids p.
Il nous faut montrer que p < dmin(C)) l'erreur est détectable or l'erreur est détectable si et seulement si y ∉ C. Il nous faut donc montrer que : p < dmin(C) ⇒ y ∉ C
Nous allons montrer la contraposée : y ∈ C ⇒ p ≥ dmin(C)
Nous supposons donc maintenant que y est dans C. Comme il y a une erreur d (x, y) 6= 0 ; de plus x est dans C. Donc, par définition de la distance minimale, p = d (x, y) <dmin(C).

(2) - Soit x le mot transmis et y le mot reçu à travers un canal à erreurs. Soit p le poids de l'erreur, donc d(x, y) = p et, par hypothèse, p < $\frac{d_{min}(C)}{2}$. Nous connaissons

y mais pas x ; nous ne connaissons pas p non plus, mais nous savons que p < $\frac{d_{min}(C)}{2}$. Pour corriger l'erreur, nous cherchons un mot qui soit à distance de y inférieure à $\frac{d_{min}(C)}{2}$. Il en existe au moins un, par hypothèse, le mot x transmis. Nous allons montrer par l'absurde qu'il n'en existe pas d'autre. En effet, soit x0 un deuxième mot possible. Nous avons, par construction :

d(x, y) < $\frac{d_{min}(C)}{2}$ d(y, x0) < $\frac{d_{min}(C)}{2}$

donc par l'inégalité triangulaire : d(x, x0) ≤d(x, y) + d(y, x0) < dmin(C)
Or x et x0 sont deux mots de code distincts, donc dmin(C) ≤ d(x, x0), ce qui est une contradiction.
Donc nous avons montré que pour tout autre mot de code x0, d(x0, y) ≥ $\frac{d_{min}(C)}{2}$.
Donc d (x, y) < d (x0, y) et x est le mot de code le plus proche de y.

Les sens réciproques de (1) et (2) se montrent par contraposée et par l'absurde.

(4) - Posons $\delta = d_{\min}(C)$. Soit f l'application qui, à un mot de code x, associe le mot obtenu en supprimant les $\delta - 1$ derniers symboles. C'est donc une application $C \rightarrow A^{n-\delta+1}$.

Montrons que f est injective. Soient deux mots de code $x \neq x'$; montrons par l'absurde que $f(x) \neq f(x')$. Supposons que $f(x) = f(x')$, alors x et x' ne peuvent différer que dans leur $\delta - 1$ derniers symboles, donc $d(x, x') \leq \delta - 1$, ce qui contredit la définition de la distance minimale. Donc, par le principe de l'injection, $\text{card}(C) \leq \text{card}(A^{n-\delta+1}) = [\text{card}(A)]^{n-\delta+1}$ En prenant le logarithme à base card(A) nous obtenons $\log_{\text{card}(A)} \text{card}(C) \leq n - \delta + 1$ or le terme de gauche vaut k , d'où le résultat voulu.

Lemme : Soit B une boule de cardinal q ($|B| = q$) et **x ∈ C,** alors pour 0≤e≤n, $|B\text{e}(x)| = \sum_{i=0}^{e} \binom{n}{i} (q-1)^i$.

4. Autres concepts

Matrice génératrice : Une matrice génératrice pour un [n,k] - code linéaire **C** est une matrice de taille kxn dont les lignes forment une base de C. On dit que G est sous forme systématique si G est la forme $\lfloor I_K | A \rfloor$ Où ou I_k est la matrice identité de taille kxk. L'application linéaire de F_q^k dans F_q^n définie par a→ aG est l'opération d'encodage. Ainsi, si G est sous sa forme systématique et si on veut encoder m ∈ F_q^k on calcule mG=[m,mA]; les k premiers bits sont alors les bits d'information et les n-k bits suivants sont les bits de redondance c'est-à-dire les bits contenant des erreurs e.

Remarque : Si $(v_i)_{1\leq i\leq k}$ est une base de C, elle est obtenue en écrivant sur les lignes chaque v_i. L'ordre importe peu car par une permutation de lignes, on obtient toujours une matrice génératrice qui lui est équivalente.

Exemple : pour q=2, les vecteurs v_1(0,0,1,1,1,0,0) ; v_2(0,1,1,1,0,1,1) ; v_3(1,1,1,0,1,0,0) forment une base du code et on a comme possible matrice génératrice :

$$\mathbf{G} = \begin{bmatrix} \mathbf{0\,0\,1\,1\,1\,0\,0} \\ \mathbf{0\,1\,1\,1\,0\,1\,1} \\ \mathbf{1\,1\,1\,0\,1\,0\,0} \end{bmatrix}$$

Matrice de parité ou de contrôle: Le code dual de C est le [n,n-k]-code linéaire C^T défini par : $C^T=\{v \in F_q^n \mid u.v=0, \text{ pour tout } u\in C\}$. Une [n,n-k] matrice génératrice H de C^T est appelée matrice de parité du code C. La matrice de parité caractérise bien C car on a la relation $C=\{m\in F_q^n \mid Hm^T=0\}$.

Syndrome : soit x le mot de code transmis et y le mot de code reçu. On appelle syndrome le vecteur donné par $\mathbf{s = yH^t}$. Il permet de détecter des erreurs mais pas toutes (cas où l'erreur est un mot de code). En effet, il n'y a pas d'erreurs transmis si $\mathbf{s = 0}$ et dans le cas contraire, il y'a eu une erreur. Lorsque le syndrome est différent du vecteur nul, il est identique à la colonne de contrôle correspondant au bit à changer.

II. LES CODES CORRECTEURS D'ERREUR

A. Définition de base

Un **code correcteur**, souvent désigné par le sigle anglais **ECC** (de l'anglais : *error-correcting code*), aussi appelé **code correcteur d'erreur(s)** ou **code de correction d'erreur(s)** (**CCE**)1, est une technique de codage basée sur la redondance. Elle est destinée à corriger les erreurs de transmission d'une information (plus souvent appelée message) sur un canal de communication peu fiable. Cela est dû à différents bruits ou anomalies météorologiques et est traduit par un effacement de bits ou un changement de bits La théorie des codes correcteurs ne se limite pas qu'aux communications classiques (radio, câble coaxial, fibre optique, etc.) mais également aux supports de stockage comme les disques compacts, la mémoire RAM et d'autres applications où l'intégrité des données est importante.

B. Principe général

Chaque suite de bits à transmettre est augmentée par une autre suite de bits dite « de
Redondance » ou « de contrôle ». Pour chaque suite de k bits transmise, on ajoute r bits à travers processus d'encodage où l'encodeur est une fonction linéaire. On

dit alors que l'on utilise un code C(n, k) avec n = k + r . À la réception, les bits ajoutés permettent d'effectuer des contrôles.
Parmi, les techniques de contrôle de redondance, on peut citer :

La somme de contrôle (en anglais « checksum ») est un cas particulier de contrôle par Redondance. Elle permet de détecter les erreurs, mais pas forcément de les corriger. En effet, Les codes utilisant les sommes de contrôle permettent de valider un message. Si le nombre d'altérations durant la transmission est suffisamment petit, alors elles sont détectées. L'utilisation d'une unique somme de contrôle permet la détection mais non la correction des erreurs. Le principe est d'ajouter aux données des éléments dépendant de ces dernières et simples à calculer. Cette redondance accompagne les données lors d'une transmission ou bien lors du stockage sur un support quelconque permettant de vérifier, avec une très haute probabilité, que l'intégrité de ce bloc a été préservée lors d'une opération de copie, stockage ou transmission. On parle aussi parfois d'**empreinte numérique**. Pour l'utilisateur final, les sommes de contrôle se présentent typiquement sous la forme de nombres au format hexadécimal. Plus tard, il est possible de réaliser la même opération sur les données et de comparer le résultat à la somme de contrôle originale, et ainsi conclure sur la corruption potentielle du message.

EXEMPLE SIMPLE :

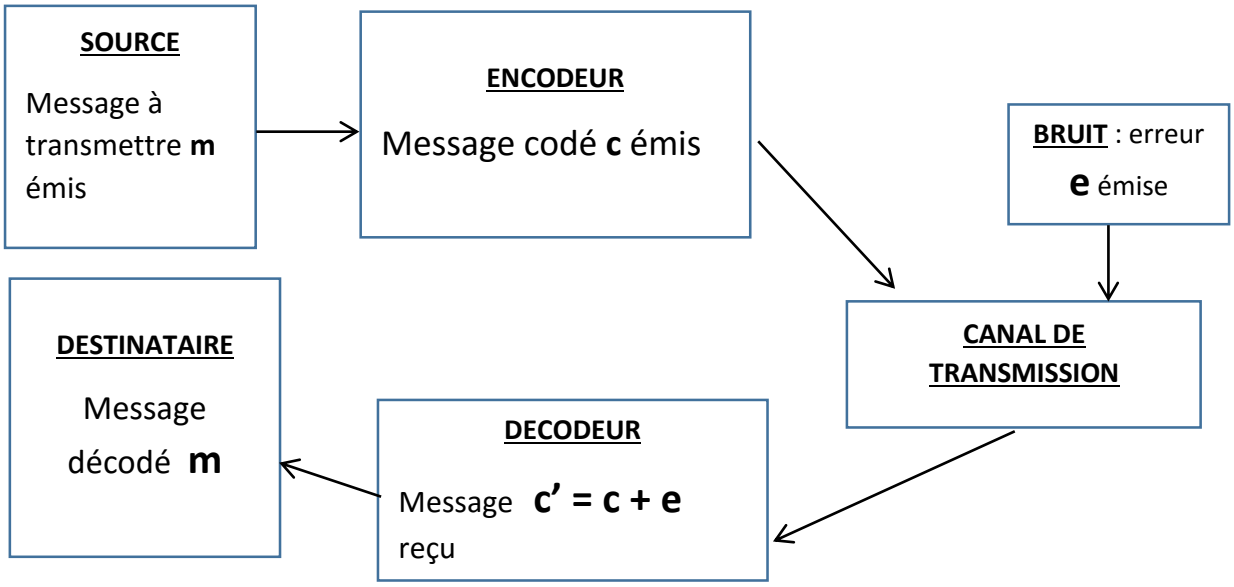
On présente ici un exemple élémentaire de code correcteur obtenu en complétant une suite de trois nombres (constituant l'information à transmettre) par deux autres nombres (constituant le code de contrôle de l'information). L'ensemble des cinq nombres permet alors de détecter et de corriger une erreur qui se serait produite sur l'un des trois premiers nombres lors de la transmission.
Soit donc un bloc de 3 nombres que l'on souhaite transmettre : 02 09 12
Ajoutons deux nombres de contrôle de l'information.
Le premier est la somme des 3 nombres : 02 + 09 + 12 = 23
Le second est la somme pondérée des 3 nombres, chacun est multiplié par son rang : 02×1 + 09×2 + 12×3 = 56
À la sortie du codeur, le bloc à transmettre est : 02 09 12 23 56
À la suite d'une perturbation, le récepteur reçoit : 02 13 12 23 56
À partir des données reçues, le décodeur calcule :
Sa somme simple : 02 + 13 + 12 = 27
Sa somme pondérée : 02×1 + 13×2 + 12×3 = 64
La différence entre la somme simple calculée (27) et celle reçue (23) indique la valeur de l'erreur : 4 (27-23 = 4)

La différence entre la somme pondérée calculée (64) et celle reçue (56), elle-même divisée par la valeur de l'erreur indique la position où l'erreur se trouve : 2 ((64-56) / 4 = 2).
Il faut donc retirer 4 au nombre du rang 2.
Le bloc original est donc 02 (13-4=09) 12 23 56

Bit de parité : C'est la forme la plus simple de détection d'erreur. Un bit de parité est un bit que l'on ajoute au début d'une suite de bits pour nous informer de la parité du nombre de bits allumés (bit égal à 1) ou nombre bits tout court selon le formalisme utilisé.
Exemple simple : Transmettons sept bits auxquels viendra s'ajouter un bit de parité. On peut définir le bit de parité comme étant égal à zéro si la somme des autres bits est paire et à un dans le cas contraire. Si la somme des bits est impaire, c'est qu'il y a eu une erreur de transmission.
1010001 (7 bits) devient 10100011 (8 bits).

En observant les concepts utilisés, les messages à transmettre ont une longueur fixe or dans le cas général, les messages à transmettre n'ont pas de longueur fixe. Cette situation existe, par exemple, pour une communication téléphonique. En revanche, il est plus simple de développer un code correcteur pour des messages d'une longueur fixe. La solution utilisée consiste à segmenter la difficulté. Dans un premier temps, est traité le cas d'un message de longueur fixe. Pour le cas général, une solution simple consiste à concaténer une suite de blocs. **Un code en bloc** est un code correcteur traitant des messages de longueur fixe.

Une vue graphique des codes correcteurs d'erreurs est illustrée par la figure ci-dessous :



Comme on peut le constater en observant cette figure, des problématiques liés à la détection d'erreurs, à la correction d'erreurs et à l'efficacité d'un code correcteur d'erreurs sont soulevées. En effet, dans la pratique, on ne connait pas le mot reçu pour faire l'analyse.

C. Les problèmes du décodage

■ La détection d'erreur

Supposons par exemple que l'on arrive à prouver que le destinataire a reçu un message qui n'est pas un mot du code. Il apparait donc qu'une (ou plusieurs erreurs) s'est (se sont) glissées dans le message au cours de la transmission. On dit donc que l'on a détecté la présences d'erreurs. Si aucune erreur n'a été détectée, on conclut que l'on a soit reçu un mot du code, soit l'on a reçu un mot qui comportait trop d'erreurs. Les propositions énoncées au I.B.3 nous permettent de connaitre le poids maximal d'erreurs qu'un code peut détecter.

■ Correction d'erreur

Détecter c'est bien, corriger c'est mieux. Dans le cas d'un canal à erreurs, la correction est plus difficile, car on ne sait pas à quelles positions il y a des erreurs. Une méthode simple consiste à chercher le mot x le plus proche du mot y reçu. Les propositions énoncées au I.B.3 nous permettent de connaitre le poids maximal d'erreurs qu'un code peut corriger.

Dans la suite, nous présenterons quelques exemples de codes correcteurs d'erreur pour mettre en évidence les techniques employés.

D. Quelques codes correcteurs d'erreur

a) Code de Hamming

Les codes de Hamming sont des codes correcteurs d'une seule erreur particulièrement simple du point de vue du codage et du décodage. Nous les présenterons comme des codes linéaires sur F_2. Ces codes satisfont l'égalité de Hamming. Pour réaliser de tel code, on prend donc la distance minimale la plus petite possible, dmin=3. Posons, r = n – k, $r \geq 2$ *un entier*.

Un **code de Hamming** est un code de paramètres $(2^r - 1, 2^r - r - 1, 3)$ dont une matrice de contrôle est obtenue par n'importe quelle énumération en colonne de tous les mots de m bits non nuls.

Principe du codage

On fixe un entier r, redondance, et on construit H une $\times 2^r - 1$ matrice. H est une matrice appelée matrice de contrôle du code de Hamming avec r redondances. Ce code est noté Ham(r). Ham(r) a pour longueur $n = 2^r - 1$ et pour dimension k = n – r. Il comprend donc r = n – k bits de redondance.

Exemple : Etudions le cas r = 3.

Considérons un code linéaire systématique C (7, 4, 3). Soit $x \in \mathbf{F^4}$, x = d1d2d3d4 et soit f : $\mathbf{F^4} \rightarrow F_2^7$

Telle que f(x) = d1d2d3d4p1p2p3 avec p1 = d1 + d2 + d4, p2 = d1 + d3 + d4 et p3 = d2 + d3 +d4.

On obtient la matrice génératrice

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

Encodons un mot, modifions un de ses bits et décodons-le,

Soit m = (1010). On a f(m) =1010101

Modifions le 4^e bit (1010101) on obtient (1011101). Comme le code est systématique, alors La matrice de contrôle de H nous donne :

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

b) Codes de Reed-Muller

Il s'agit d'un algorithme permettant de fabriquer facilement des codes. En particulier ces codes linéaires, ayant des caractéristiques intéressantes en termes de rapport dimension/longueur et de pouvoir de correction. En effet, la distance minimale d'un code de Reed-Muller est égale à la moitié de la longueur du mot.

Principe de codage

Soit un code de Reed-Muller d'ordre m et de longueur 2^r ou $0 \leq m \leq r$.

Définition intuitive

Noté RM (m, r) il est défini par : RM (0, r) = {00...0, 11...1}

RM (r, r) = $\{0, 1\}^{2\wedge r}$

RM (m, r) = {(x, x+y) :x dans RM(m,r-1), y dans RM(r-1,m-1)}

Matrice génératrice du RM(m,r)

Elle est notée G (m, r) 0<m<r, on a :

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

- m=0, on définit la matrice 1×2^r :

G (0, r) = [11…1]

- Si m=r, on obtient la matrice : $\mathbf{G(r,r)}=\begin{pmatrix} \boldsymbol{G(r-1,r)} \\ \mathbf{0 \ldots \ldots 1} \end{pmatrix}$

Exemple :

$$G(1,3)=\begin{pmatrix} G(1,2) & G(1,2) \\ 0 & G(0,2) \end{pmatrix}=\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{pmatrix}$$

G(0,2)=(1 1 1 1) et $G(1,2)=\begin{pmatrix} G(1,1) & G(1,1) \\ 0 & G(0,1) \end{pmatrix}=\begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}$

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

Propriétés

Le code de Reed-Muller RM (1,m, r) présente les propriétés suivantes

- Longueur $n=2^r$
- Distance $d=2^{r-m}$
- Dimension $k=\sum_{i=0}^{m}\binom{r}{i}$
- RM (r-1, m) est contenu dans RM (m, r) Comme code dual, RM (r-1-m, r) $m \leq r$.

c) Code de Reed-Solomon

Le problème du décodage est difficile pour les codes binaires, c'est-à-dire, pour les codes sur F_2. Mais si nous considérons un alphabet plus grand, une solution optimale a été proposée dans les années 1950 par Irvine Reed et Gus Solomon. Ces codes sont appelés les codes de Reed-Solomon en leur honneur et jusqu'à aujourd'hui ce sont les codes les plus utilisés. Chaque seconde, il y a des

centaines de millions des codes Reed-Solomon en fonctionnement, et ils assurent que la plupart de nos communications soient essentiellement sans erreurs. Même si les codes Reed-Solomon sont puissants, leur définition est simple ; elle est basée sur des évaluations de polynômes.

Définition (Reed-Solomon) Soient n et k des entiers avec $1 \leq k \leq n$. Un code de Reed Solomon de paramètres (n, k) est défini comme suit :

1. L'alphabet est un corps fini K de cardinal $\geq n$.
2. Choisissons n éléments distincts de K, $a_1, a_2..., a_n$. Une suite de k symboles $u = (u_1, ..., u_k) \in K^k$ est encodée en la suite de n symboles $x= (x_1, ..., x_n) \in K^n$ définie par $x_i = P_u(a_i)$ pour i = 1...n

   Le code de Reed-Solomon C'est l'ensemble de tous les encodages x possible pour tous les $\rightarrow u \in K^k$. C'est donc un code linéaire en bloc de longueur (taille) n et de dimension k.

Exemple (Code de Reed-Solomon sur F_5) Construisons un code de Reed-Solomon sur F_5. Nous pouvons choisir n'importe quelle longueur n entre 1 et 5 = card(F_5). Choisissons le maximum possible, n = 5.

Il nous faut aussi choisir 5 éléments de F_5, mais ici nous n'avons pas le choix, il faut les prendre tous : $a_1 = 0$, $a_2 = 1$, $a_3 = 2$, $a_4 = 3$ et $a_5 = 4$.
Calculons les encodages. Par exemple, si le message est u = (0, 0), le polynôme à évaluer est $P_0 = 0$ donc $P_0(a_1) = ... = P_0(a_5) = 0$, et le mot de code correspondant est x = (0, 0, 0, 0, 0).
Si u = (3, 2), alors $P_u(X) = P_{32}(X) = 3 + 2X$ et donc :
$P_u(a_1) = P_u(0) = 3$.
$P_u(a_2) = P_u(1) = 0$
$P_u(a_3)=P_u(2)=2$
$P_u(a_4) = P_u(3) = 4$
$P_u(a_5) = P_u(4) = 1$

Donc x = (3, 0, 2, 4, 1). En faisant cela pour les 25 suites de k = 2 éléments de F_5 nous obtenons la table d'encodage de la Figure ci-dessous :

U	$P_U(x)$	x	

00	0	00000	31	3 + X	34012
01	X	01234	32	3 + 2X	30241
02	2X	02413	33	3 + 3X	31420
03	3X	03142	34	3 + 4X	32104
04	4X	04321	40	4	44444
10	1	11111	41	4 + X	40123
11	1 + X	12340	42	4 + 2X	41302
12	1 + 2X	13024	43	4+3x	42031
13	1 + 3X	14203	44	4+4x	43 210
14	1 + 4X	10432			
20	2	22222			
21	2 + X	23401			
22	2 + 2X	24130			
23	2 + 3X	20314			
24	2 + 4X	21043			
30	3	33333			

Une matrice génératrice de ce code de Reed-Solomon sur F_5 est obtenue en considérant les encodages de la base canonique de F_5^2. La première ligne est obtenue est obtenue en prenant u = (1, 0), ce qui correspond au polynôme P_{10} = 1 ; la deuxième ligne est obtenue est obtenue en prenant u = (0, 1), ce qui correspond au polynôme P_{01} = X. Donc :

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ a_0 & a_1 & a_2 & a_3 & a_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \end{pmatrix}$$

Le mot de code correspondant au message u = (u1, u2) est x = (u1, u2) G.

Les codes de Reed-Solomon sont optimaux en terme de distance minimale, et sont utilisés sur un très grand nombre de systèmes. Par exemple, les lecteurs de CDs utilise plusieurs codes de Reed-Solomon imbriqués. La correction d'erreur est une tâche plus difficile. Ici, la linéarité aide beaucoup et il faut aussi des structures supplémentaires. Mais ceci est une autre histoire... Il est maintenant grand temps d'attaquer le thème de notre sujet d'étude : Le principe du décodage par ensemble d'information (En anglais :***Information Set Decoding en abregé ISD***) et ses algorithmes.

III. DECODAGE PAR ENSEMBLE D'INFORMATION

Lors de la grande partie précédente, on remarque que la recherche de mots de faible poids de Hamming dans un code linéaire est un problème important en théorie des codes et en cryptographie. Une famille d'algorithmes permettant de résoudre ce problème suit une approche de décodage par ensemble d'informations ; ce sont généralement les algorithmes les plus efficaces quand la longueur du code est proportionnelle à sa dimension. Le décodage par ensemble d'informations réside sur le principe de la recherche des positions I de C n'ayant pas d'erreurs. I, C seront définis par la suite.

A. Définition

Soit C un (n,k ,d)-code, G sa matrice génératrice.

Quantité d'information : Procédons par une explication par un exemple : Considérons un texte dense en contenu. Il est difficile de le remplacer par un texte plus court qui contient la même information. Pour mieux d´développer ce point de vue, imaginons que nous ayons réussi `à condenser au maximum l'information contenue dans un texte sous la forme (typique en informatique) d'une suite de 0 et de 1. Chacune de ces unités d'information est appelé un bit. Le nombre de ces bits serait alors considéré comme donnant la *quantité d'information* contenue dans le texte en question.

Ensemble d'information : Étant donné un sous-ensemble $I = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ et un mot $X \in F_q^n$, on note $X_i = (x_1, \ldots, x_k) \in F_q^k$. Pour un code $C \subseteq F$, son poinçonnement en dehors de I (c'est-à-dire sa restriction à I) est : $\mathbf{C_I = \{c_I, c \in C\}}$ **.** Soit $C \subseteq F_q^n$ un code de dimension *k*. Un ensemble $I \subseteq \{1, \ldots, n\}$ de cardinal k est un ensemble d'information pour C si $C_i = F_q^k$ c'est-à-dire si $\mathbf{G}_I$ est inversible . Il n'y a pas unicité de l'ensemble d'information d'un code. Par exemple, le code binaire engendré par $G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$ possède comme ensembles d'information {1, 2,3}, {4, 5,7} et d'autres.

Proposition : Un code MDS (*i.e.* un code de distance minimale $n - k + 1$) admet comme ensembles d'information tous les sous-ensembles de taille *k* de $\{1, \ldots, n\}$.

Comme l'on peut s'en douter, il existe de nombreuses variantes du décodage par ensemble d'information. Cette pléthore de variantes de l'ISD permet d'en réduire le facteur de travail. Ces divers algorithmes peuvent paraitre à première vue hétéroclites (qui s'écartent des règles) ; en fait, ils exploitent principalement deux idées. La première c'est d'autoriser certains émotifs d'erreur dans l'ensemble d'information. C'est cette technique introduite par KASAMI dans certains codes correcteurs La deuxième idée consiste dans un premier temps à décoder jusqu'à a une certaine distance du code poinçonné obtenu en enlevant certaines positions de redondance du code C puis à vérifier si le décodage partiel s'applique au code tout entier.

B. Historique

- Le Décodage Par Ensembles D'informations a été créé par PRANGE en 1962.
- Il a été utilisé par McELICE en 1968 pour la cryptanalyse de son crypto système.
- Il a été revisité par LEE-BRICKELL qui, pour diminuer la complexité a augmenté le nombre d'opération par itération.
- LEON utilise la même approche que LEE-BRICKELL mais cette fois si en utilisant un ensemble de redondance *L*.
- STERN utilise le même principe que LEON mais cette fois si en divisant l'ensemble d'information en 2 parties.
- DUMER utilise un principe similaire à STERN sauf que le sien reduit le coût de l'algorithme.
- L'algorithme de BECKER, JOUX, MAY et MEURER: Cet algorithme a été introduit en 2012 comme une des variantes du décodage par ensemble d information.
- L'algorithme de MAY, MEURER et THOMAE a été introduit en 2011: On garde la même approche que Dumer mais cette fois, on utilise une méthode appelée <<**Association des colonnes**>>.

C. Principe general du décodage par ensemble d'information

On note G comme étant une décomposition en (U,V) par rapport à I c'est-à-dire que $U=(Gi)_{i \in I}$ et $V=(Gj)_{j \in J}$ où $(G_1,...,G_n)$ est la décomposition en colonne de la matrice génératrice G. La définition originelle de I (ensemble d'informations) se fait en supposant qu'un mot de code C=mG+e soit sous la forme décomposée $(C_I,C_J)=m(U|V)+(e_I|e_J)$ avec $e_I = C_I - mU$ et $e_J=c_J-mV$. Si I ne contient pas d'erreurs, alors on aura $c_I = mU$ et $e_J=c_J-mV$; cette opération est valable tant que U n'est pas singulière (non inversible). Ainsi donc, des lors que l''on trouve un ensemble d'information I ne contenant pas de position d'erreurs, il suffit de décomposer) et $C=(C_I,C_J)$ et $G=(G_I,G_J)$ pour obtenir le vecteur erreur. En effet, on a C=X+e, où X=mG, donc $mU = X_I$ $X=mG= X_IU^{-1}G$. Or $c_I = Xi$, car I ne contient aucune position d'erreurs. Par conséquent $e=C-X=C- c_I U^{-1}G=(0.....0, C_J - c_I U^{-1}V)I$ et $m= c_I U^{-1}$. C désignant le mot à décoder formé de la somme d'un mot du code X et du vecteur d'erreur e de poids minimal t, L'algorithme repose sur le fait que, tant qu'on n'a pas trouvé un vecteur d'erreurs de poids inférieur ou égal à t, on choisit I, on calcule $Z= U^{-1} V$,et décompose c sous la forme (C_I,C_J),

enfin on calcule le poids minimal w(cJ- cIZ). Si ce poids est inferieur a t, alors $e=(0.....0,C_j - C_iU^{-1}V)$ et $m= c_I U^{-1}$.

D. Quelques algorithmes de décodage par ensemble d'information

1. Algorithme de LEE-BRICKELL

Il consiste simplement à rechercher un ensemble d'information contenant au plus P positions erronées. (P est un paramètre de l'algorithme dont la valeur optimale est 2 pour l'algorithme originel). Il peut être considéré comme un algorithme de décodage par ensemble d'information avec p effacements. Cette approche est justifiée par le fait que, dans le décodage par ensemble d'information, l'élimination de Gauss permettant de mettre la matrice systématique est de loin la procédure la plus coûteuse. Afin d'accéder à un meilleur compromis entre le nombre d'itérations et le coût de chacune d'elles, il est donc préférable d'augmenter légèrement le facteur de travail de chaque itération en vérifiant en plus si l'ensemble d'information considéré ne contient pas qu'un tout petit nombre de positions erronées (effacement), car celui-ci réduit considérablement le nombre d'itération.

Algorithme :

Début :

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à t n'a pas été trouvé:

-Choix aléatoire de I

-On calcule $Z=U^{-1}V$, et décompose c sous la forme$(c_I,c_J)_I$

-Pour tous les vecteurs e'_i de longueur k et de poids $i \leq p$, calcul de $wt(c_J-(c_I-e'_i)Z$

Si ce poids est inférieur ou égal à t-i, alors $e=(e'_i,c_J-(c_I-e'_i)Z)_I$ et $m=(c_I-e'_i)U^{-1}$

Fin

Complexité (facteur de travail) :

La probabilité pour qu'un ensemble d'information contienne au plus p positions d'erreur est $Q_p = \frac{1}{\binom{n}{k}} \sum_{i=0}^{p} \binom{n-t}{k-i}\binom{t}{i}$

Soit Np le nombre de vecteurs d'erreur de k bits avec au plus p bits non nuls, alors $Np = \sum_{i=0}^{p} \binom{k}{i}$

Par conséquent, le facteur total de travail de l'algorithme de Lee-Brickell avec paramètre p est $Wp = \frac{1}{Qp}(\alpha k^3 + N_p \beta k)$ où β est une constante.

2. L'algorithme de STERN

Algorithme :

Début :

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à t n'a pas été trouvé :

- Choisir aléatoirement un ensemble d'information I et un ensemble L de l positions de redondance.
- A l'aide d'une élimination de Gauss, mettre G sous la forme suivante $U^{-1}G = (Id_k|Z_L|Z_{J\setminus L})$ et décomposer c sous la forme $(c_I|c_L|c_{J\setminus L})$.
- Diviser aléatoirement les LIGNES de la matrice $(Z_L|Z_{J\setminus L})$ en deux parties de même taille.
- Pour tous les vecteurs e'_1 de longueur $\left\lfloor \frac{k}{2} \right\rfloor$ et de poids p, calculer $e'_1 Z_L^1$.
- Pour tous les vecteurs e'_2 de longueur $\left\lfloor \frac{k}{2} \right\rfloor$ et de poids p, calculer $e'_2 Z_L^2$.
- Pour tout couple (e'_1, e'_2) tel que $c_L = c_I Z_L - e'_1 Z_L^1 - e'_2 Z_L^2$, calculer $wt(c_{J\setminus L} - c_I Z_{J\setminus L} + e'_1 Z_{J/L}^1 + e'_2 Z_{J/L}^2) = wt(c_{J\setminus L} - (c_I - (e'_1|e'_2))Z_{J\setminus L})$.
- Si ce poids est inférieur ou égal à $t - 2p$, alors
- $e = (e'_1|e'_2|0 \cdots 0|c_{J\setminus L} - (c_I - (e'_1|e'_2))Z_{J\setminus L})$ et $m = (c_I - (e'_1|e'_2))U^{-1}$.
- Fin

Complexité :

- La probabilité que I contienne 2p positions d'erreurs est $Q_{S1} = \frac{\binom{t}{2p}\binom{n-t}{k-2p}}{\binom{n}{k}}$.
- La probabilité que I_1 contienne p positions d'erreurs parmi ces 2p est $Q_{s2} = \frac{\binom{2p}{p}}{4p}$ (car $\sum_{i=0}^{2p}\binom{2p}{i} = 2^{2p} = 4^p$).
- La probabilité que L ne contienne aucune position d'erreurs est QS3 $= \frac{\binom{n-k-t+2p}{l}}{\binom{n-k}{l}}$.

Ainsi la probabilité que le vecteur d'erreur corresponde au motif est

$Q_{S(p,l)} = Q_{S1} \times Q_{S2} \times Q_{S3} = \frac{\binom{t}{2p}\binom{n-t}{k-2p}\binom{2p}{p}\binom{n-k-t+2p}{l}}{4p\binom{n}{k}\binom{n-k}{l}}$

Le facteur total de travail de l'algorithme de Stern est donc de $W_{S(p,l)}=\frac{1}{QS(p,l)} * G(p,l)$, où G(p, l) est une fonction des deux paramètres représentant le nombre d'opérations de chaque itération de l'algorithme.

3. L'algorithme de LEON

L'algorithme proposé par J.S. Léon était initialement conçu pour la recherche de mots de poids minimal dans un code linéaire mais il peut également être appréhendée comme un algorithme de décodage. La seule différence qu'il introduit par rapport au décodage par ensembles d'information avec effacement est qu'il considère `à chaque itération un ensemble de positions S de taille supérieure `à k qui, de plus, ne contient pas nécessairement un ensemble d'information. Mais, tout comme l'algorithme de Lee-Brickell, il décode tous les motifs d'erreurs dont la restriction à S est de poids inférieur ou égal à p. Les Motifs d'erreurs corrigés à chaque itération par l'algorithme de Léon (version originale) : on corrige tous les vecteurs d'erreurs dont la restriction à la sélection S est de poids au plus p.

Algorithme :

Entrée et sortie : sont les mêmes que celles de l'algorithme de Prange.

Début :

Tant qu'un vecteur d'erreurs de poids inférieur ou égal à t n'a pas été trouvé :

- Choix aléatoire de I et un ensemble L dit filtre
- A l'aide d'une élimination de Gauss, mettre G sous la forme suivante $U^{-1}G =(Id_k|Z_L|Z_{J\setminus L})$ et décomposer c sous la forme $(c_I |c_L|c_{J\setminus L})$.
- Pour tous les vecteurs e'_i de longueur k et de poids $i \leq p$, calculer $t'= wt(c_L - (c_I - e'_i)Z_L)$.

– Si $t' \leq p - i$, calculer $wt(c_{J\setminus L} - (c_I - e'_i)Z_{J\setminus L})$.

– Si de plus ce poids est inférieur ou égal à $t - t' - i$, alors

$e = (e'_i |c_L - (c_I - e'_i)Z_L|c_{J\setminus L} - (c_I - e'_i)Z_{J\setminus L})$ et $m = (c_I - e'_i)U^{-1}$

Fin

Complexité :

La probabilité pour que l'ensemble S = I∪L contienne au plus p positions d'erreur est

$Q_{L(p,l)} = \frac{1}{\binom{n}{k+l}} \sum_{i=0}^{p} \binom{n-t}{k+l-i}\binom{t}{i}$.

Ainsi le facteur de travail de l'algorithme de Leon est $WL(p,l) = \frac{1}{QL(p,l)} \times F(p,$ l), où F(p, l) est une fonction des deux paramètres représentant le nombre d'opérations de chaque itération de l'algorithme.

On retrouve $W_L(p,0) = W_p$ car si on ne considère pas l'ensemble L, on effectue exactement l'algorithme de Lee-Brickell.

4. L'algorithme de MAY, MEURER et THOMAE

Principe : C'est d'essayer de résoudre le décodage par syndrome (Q,s,p) en construisant un groupe de mots données par $e_1,e_2 \in F$ tel que $wt(e_1)=wt(e_2)=p^2$.

Algorithme : Association_colonne

Entrée : $Q \in F^{lxk+1}$ 2 , $s \in^{f*l}$, $s \in F^{n-k*2}$, $0 \leq p \leq k + l$

Sortie : $L \subset \{e \in F^{l*2} ; wt(e) \leq p, Qe = s\}$

Paramètres : $L_1, L_2 \subset [l]$, L_1] $L_2 = L$ et $|Li| = li$, pour i = 1, 2.

```
 Répéter
   Construire L1,1, L1,2, L2,1 et L2,2
      Arranger la liste L1,1 et L2,,2  en respectant le deuxième élément.
         Pour chaque (e1,1, QL2 e1,1) ∈ L1,1, faire
              Pour chaque (e1,2, QL2 e1,2) ∈ L1,2 faire
                   Si QL2 e1,1 = QL2 e1,2 alors
                     Faire e1 = e1,1 + e1,2
                     Ajouter (e1, QL1 e1) à L1
                        Pour chaque (e2,1, QL2 e2,1) ∈ L2,1, Faire
                             Pour chaque (e2,2, QL2 e2,2 + sL2 ) ∈ L2,2, faire
                                 if QL2 e2,1 = QL2 e2,2 + sL2 alors faire  e2 = e2,1
+ e2,2
                                   Ajouter (e2, QL1 e1 + sL1 ) à L2
                                   Arranger la liste L2 en préservant le deuxième
composant
         Pour chaque (e1, QL1 e1) ∈ L1, faire
           Pour chaque (e2, QL1 e2 + sL1 ) ∈ L2, faire
          Si QL1 e1 = QL2 e2 + sL1  nous  ajoutons e1 + e2 à L
 Retourner L
```

Décodage de May, Meurer et Thomae

Entrée : $H \in F^{n-k*2}$, $s \in F^{n-k*2}$, $s \in F^{n-k*2}$, $w \in Z$, $w \geq 0$

Sortie : $e \in F^n$ tel que $He = s$ et $wt(e) = w$

Paramètres p, l et l1, l2 tel que l = l1 + l2
Répéter
Choisir une matrice aléatoire $P1 \in F^{nxn*2}$
Calculer $P2 \in F^{n*n}$ et $U \in F^{n-k*lxn-k*l}$ tel que $H_o = UHP1P2$
Calculer s' = Us,
Nous avons I = [n − k − l + 1, n] et J = [n − k − l + 1, n − k].
Calculer L = **Association_Colonne**
Pour chaque e ∈ L Si wt(Qe + s 0) = w − wt(e)
Retourner P1P2(Qe + s 0 , e)

5. L'algorithme de BECKER, JOUX, MAY et MEURER

Maintenant nous présenterons une autre évolution du décodage par ensemble information. L'idée : Nous augmentons la valeur d'un paramètre epsilon jusqu'à w2/4n.

Algorithme : Décodage de Becker, Joux, May et Meurer

Entrée: $H \in F^{n-kxn}$, $s \in F^{n-kx2}$, $s \in F^{n-kx2}$, w ∈ Z positive
Sortie: $e \in F_n{}^2$ tel que He = s et wt(e) = w
Paramètre p, l et p1 ∈]0, p/2[, p2 ∈]0, p1/2[
Choisir une matrice $P1 \in F^{nxn-2}$
Calculer P2 ∈ F n×n 2 et $U \in F^{n-k-ixn-k-i}$ tel que
H' = UHP1P2 = $(Id_{n-k}|Q)$
Calculer s' = Us,
On fait I = [n − k − l + 1, n] et J = [n − k − l + 1, n − k].
Calculer L = Association_Colonne(QJ , s' I , p, p1, p2).
Pour chaque e ∈ L faire wt(Qe + s 0) = w − p
Retourner P1P2(Qe + s , e)

E. Quelques applications du décodage par ensemble d'information

Les principaux domaines d'application du décodage par ensemble d'information sont la télécommunication, le stockage et le partage des données :

- Dans la telecommunication:

Comme nous l'avons dit mainte fois, le décodage par ensemble d'information permet de corriger dans la limite du possibles les erreurs introduits dans un message ou signal lors de la transmission.

- Dans le stockage des données :

Les codes correcteurs dont le décodage par ensemble d'information fait partie sont souvent utilises pour améliorer la fiabilité des supports de stockage des données. Les codes de Reed-Solomon sont utilisés dans les disques compacts pour corriger les erreurs causes par les rayures. Les disques durs modernes utilisent les variantes des codes Reed-Solomon pour détecter les erreurs mineures dans les lectures/écritures et les corriger. Un autre exemple pour le stockage des données, est celui de la DRAM (Dynamic RAM) dont la technologie s'appuie sur les codes correcteur d'erreur notamment l'ISD.

IV. IMPLEMENTATION

ALGORITHME IMPLEMENTE : Algorithme de **Lee-Brickell**

LANGUAGE UTILISE : **PYTHON**

LIBRAIRIES LES PLUS UTILISEES :

- **NUMPY** et **SYMPY** : pour manipuler les matrices et la plupart des opérations connues les concernant (multiplication, inverse, ...). Les fonctions de ces librairies sont amplement décrites sur les sites officiels dans la bibliographie.
- **MATH** : pour manipuler les fonctions usuelles mathématiques (logarithme, factoriel, exponentiel...)

PRINCIPES DES GRANDS MODULES UTILISEES ET LES DIFFICULTES RENCONTREES :

NB : Le symbole « = » ici, traduit le type de retour de la fonction. Ce qui est entre parenthèse, le type d'entré.

MODULES	FONCTIONS	DIFFICULTES	SOLUTION
minimaldistance	dmin(liste de listes, entier, entier) = entier	Aucune	
	Werror(vecteur ligne, entier) = entier	Aucune	
	Weightx(vecteur ligne, vecteur ligne, vecteur ligne, matrice, entier, entier) = entier	Aucune	
	EIgenerator(entier, entier) = liste de listes	Aucune	
	packingradius(entier) = entier	Aucune	

	maxcapacityofcorrection(entier) = entier	Aucune	
	Maxcapacityofdetection(entier) = entier	Aucune	
	verifyMDS(entier, entier, entier) = entier	Aucune	
	Possible(liste de listes, entier, entier, entier) = booléen	Aucune	
presentation	Presentation() = tous les paramètres d'entrée	Aucune	
Generatrice	Matrice(entier, entier) = matrice	Aucune	
	Gfilling(matrice vide, entier, entier, caractère) = matrice	Agencer différentes boucles pour que les lignes entrées soit linéairement independants	Utiliser la methode **rank()** pour récupérer le rang de la matrice entrée
	Gfilling_forced(entier, entier, caractère) = matrice		
	Gfilling2(matrice vide, entier, entier) = matrice		
	Gfilling_forced2(entier, Entier		
GenerationCode	Uplet (entier) = liste de listes	Générer tous les mots du code	La fonction **product()** de la librairie **itertools**
	code(entier, matrice) = liste de listes	Aucune	
	listMod2 (liste) = liste	Opérations sur les binaires	Utiliser le **modulo 2**
	vecMod2(vecteur) = vecteur		
	llistMod2 (liste de listes) = liste de listes		
	matMod2(matrice, entier) = matrice		
	controle(matrice) = matrice	Aucune	
	syndrome(vecteur, matrice) = matrice	Aucune	
	wordincode(matrice, matrice, entier, entier) = booléen	Aucune	
Comlexityisd	complex_Prange(entier, entier, entier) : réel	Aucune	
	complex_lee(entier, entier, entier) : réel	Aucune	
	complex_leon(entier, entier, entier) : réel	Aucune	

	efficiencyalgorithm()	Aucune	
Aleatorygenerate	aleatorygenerateset(matrice, entier, entier, liste de listes) = (matrice, liste, liste, liste de listes)	Sélectionner aléatoirement les colonnes de la matrice génératrice	La fonction **random** associée à la methode **sample()** qui permet de sélectionner aléatoirement et sans répétitions des éléments d'une liste
	isIN(liste) = booléen	Complémentaire de l'ensemble d'information (J)	Utilisation de la fonction **filter()** retourne les valeurs de l'iterrable qui respectent la condition
Information_set	leebrickellISD() = entier	Aucune	
	CreateIJ(vecteur ligne, liste, liste) = (vecteur ligne, vecteur ligne)	Aucune	
	Zfixing(matrice, matrice, entier) = matrice	Aucune	
	Efixing(vecteur ligne, vecteur ligne, matrice) = vecteur ligne	Aucune	
	Rearrange(vecteur ligne, vecteur ligne) = vecteur ligne	Aucune	
exécutable		Aucune	

NB : Le principe et l'objectif de chacune des fonctions présentées n'est pas détaillé ci-dessus car le code fourni en annexe de ce document est amplement commenté pour une compréhension fluide des buts de chacune des fonctions (de plus les fonctions seront mieux expliquées lors de la présentation).

CONCLUSION

Parvenu au terme de notre analyse, il était question pour nous de présenter le principe du décodage par ensemble d'information, explorer ses différentes implémentations et de réaliser un de ses algorithmes dans un langage de programmation (En python). Pour le réaliser, nous avons tout d'abord présenter la théorie des codes correcteurs linéaires. En effet, c'est sur cette dernière que repose le principe de L'ISD. Il en ressort que le décodage par ensemble d'information est l'un des meilleurs principes de décodage basés sur les codes correcteurs d'erreur en raison de sa faible complexité et de ses multiples applications.

BIBLIOGRAPGIE

- *La cryptographie de l'Antiquité à l'Internet* Francois Bergeron et Alain Goupil 24 février 2006, Université de Québec à Montréal (UQAM).
- *Codage, cryptographie, et Applications*, Bruno MARTIN, EPFL.
- *Introduction aux codes correcteurs d'erreurs* Pierre Abbrugiati 23 janvier 2006.
- *Codes correcteurs d'erreur* Judith Louis-Alexandre, Camille Huynh, Dorian Lesbre Juin 2017.
- Codes correcteurs en cryptographie, Université de Renne 1.
- https://fr.wikipedia.org/wiki/Code_correcteur
- https://doc.sagemath.org/html/en/reference/coding/sage/coding/information_set_decoder.html
- https://www.canal-u.tv/chaines/inria/3-message-attacks-isd/35-lee-and-brickell-algorithm
- *Science de l'information*, Jean-Yves Leboudec, Patrick Thiran, RÜDIGER Urbanke , Novembre 28 2014.
- *Introduction aux codes correcteurs d'erreur, Hervé Tale Kalachi.*
- THESE de DOCTORAT de l'UNIVERSITE PARIS 6 Spécialité : Informatique présentée par Anne CANTEAUT pour obtenir le grade de DOCTEUR de l'UNIVERSITE PARIS 6 Sujet de la thèse : Attaques de crypto systèmes à mots de poids faible et construction de fonctions t-résiliente.
- http://localhost:8888/nbconvert/html/cours/CF-5.ipinb?download = false
- https://numpy.org
- https://www.sympy.org