Rapport ENS

OUADJOU Tarik

2022-2023

1 Introduction

Lors d'échanges d'informations entre un émetteur et un récepteur il se peut que le message reçu soit altéré. On s'intéresse donc dans ce rapport à l'étude des codes cycliques qui sont une famille de codes correcteurs d'erreurs qui ont pour but de corriger les erreurs subis par le message émis.

2 Défintion générale

2.1 Définition d'un code

On fixe dans ce rapport $q = p^m$ avec p premier, $m \in \mathbb{N}^*$ et $\mathbb{F}_q$ le corps à q éléments.
Une lettre a est un élément de $\mathbb{F}_q$ et un mot m de taille n est un n-uplet de lettres donc un élément de $\mathbb{F}_q^n$.
On note de plus $w(m)$ le nombre de lettres non nulles du mot m.

Définition 2.1.1 :
Un code ϕ de paramètre (n, k) est une application injective de $\mathbb{F}_q^k$ vers $\mathbb{F}_q^n$.
On dit que le code ϕ peut corriger t erreurs lorsque qu'il existe une fonction de décodage ψ tel que $\forall m \in \mathbb{F}_q^k, \forall e \in \mathbb{F}_q^n$ tel que $w(e) \leq t$ alors $\psi(\phi(m) + e) = m$. Informellement cela revient à ce que malgré t erreurs on peut retrouver le message initial.
On fera la confusion dans la suite entre l'application ϕ et l'ensemble $C = Im(\phi)$.

2.2 Distance minimale et conséquences

Définition 2.2.1 :
On appelle distance de Hamming entre 2 mots $x, y \in \mathbb{F}_q^n$ que l'on note $d(x, y)$ le nombre de lettres qui diffèrent entre x et y.
Remarque 2.2.1: $d(x, y) = w(x - y)$.

Proposition 2.2.1 :
La distance de hamming est une distance
Preuve
Il est clair que $d(x, y) = d(y, x)$, $d(x, y) \in \mathbf{R}_+$ et $d(x, y) \iff x = y$
Soit $x, y, z \in \mathbb{F}_q^n$ avec $x = (x_1, ...x_n)$; $y = (y_1, ...y_n)$ et $z = (z_1, ...z_n)$
Posons $E = \{i \in [\![1; n]\!],\ x_i \neq y_i\}$, $A = \{i \in [\![1; n]\!],\ x_i \neq y_i$ et $y_i \neq z_i\}$ et $B = \{i \in [\![1; n]\!],\ x_i \neq y_i$ et $y_i = z_i\}$
On a $|E| = d(x, y), E = A \cup B$ et $A \cap B = \varnothing$ ainsi $|E| = |A| + |B|$ et comme $|A| \leq d(y, z)$ et $|B| \leq d(x, z)$
on obtient $d(x, y) \leq d(x, z) + d(z, y)$

Défintion 2.2.2 :
On appelle distance minimale du code C $d_C = \min\{d(x, y) | x, y \in \mathbb{F}_q^n\}$.

Proposition 2.2.2 :
Le code C peut corriger toutes erreurs de poids inférieur à $e_C = \lfloor \frac{d_C - 1}{2} \rfloor$.
Preuve
Soit $y \in \mathbb{F}_q^n$ tel que y = m + e avec $m \in C$ et $w(e) \leq e_C$,
Soit $m' \in C$, supposons $d(y, m') \leq e_C$. Par inégalité triangulaire $d(m, m') \leq d(m, y) + d(y, m')$ d'ou $d(m, m') \leq 2e_C \leq d_C - 1$ par conséquent $m = m'$ par définition de d_C.
Ainsi il y a unicité de $m \in C$ tel que $d(y, m) \leq e_C$.

Remarque 2.2.1 : C ne peut pas corrigé $e_C + 1$ erreurs car il existe $x, y \in \mathbb{F}_q^n$ tel que $d(x, y) = d_C$.
Dès lors en prénant le mot z qui est x dans lequel on modifie $e_C + 1$ lettres différentes pour x et y pour les faire coïncider, on obtient $d(x, z) = e_C + 1$ et $d(y, z) = d_C - e_C - 1 \leq d_C - \frac{d_C - 1}{2} \leq \frac{d_C + 1}{2}$.
Ainsi $d(y, z) \leq \lfloor \frac{d_C + 1}{2} \rfloor$ d'ou $d(y, z) \leq e_C + 1$, ce qui conclut.

3 Code Linéaire

3.1 Défintion

Un code ϕ est dit linéaire si ϕ est une application linéaire. Dès lors ϕ étant injective on en déduit que C est un sous-espace vectoriel $\mathbb{F}_q^n$ de dimension k.
Remarque 3.1.1: $d_C = \min\{w(m) | m \in C\}$ car pour $x, y \in C$ $d(x, y) = w(x - y)$ et $x - y \in C$.

Proposition 3.1.1: Borne de Singleton
Soit C un code linéaire de paramètre (n, k) et de distance d_C alors $d_C \leq n - k + 1$.[1]
Preuve
Montrons qu'il existe $x \in C$ tel que $w(x) \leq n - k + 1$.
Posons $V = \{m = (m_0, ..m_{n-1}) \in \mathbb{F}_q^n | m_{n-k+1} = ...m_{n-1} = 0\}$, V est un $\mathbb{F}_q$ espace vectoriel de dimension $n - k + 1$. Ainsi comme $dim(V) + dim(C) > dim(\mathbb{F}_q^n)$ on a dès lors $dim(V \cap C) > 0$, d'où $V \cap C \neq \varnothing$ ce qui conclut.

3.2 Matrice génératrice et de controle

On appelle matrice génératrice du code linéaire C toute matrice G $\in \mathscr{M}_{n,k}(\mathbb{F}_q)$ tel que les colonnes forment une base de C.

Définition 3.2.1:
On appelle matrice de contrôle une matrice $H \in \mathscr{M}_{n-k,n}(\mathbb{F}_q)$ tel que $\forall m \in \mathbb{F}_q^n$ $H \times m = 0 \iff m \in C$.

On construit une telle matrice de la manière suivante : Considérons une base de l'orthogonale de C pour le produit scalaire canonique : $C^\perp = (H_1, ...H_{n-k})$.
Dès lors posons la matrice H = $\begin{pmatrix} H_1^T \\ . \\ . \\ H_{n-k}^T \end{pmatrix}$, on a $H \times m = \begin{pmatrix} H_1^T \times m \\ . \\ . \\ H_{n-k}^T \times m \end{pmatrix}$.
Ainsi $H \times m = 0 \iff \forall i \in [\![1; n-k]\!]$ $H_i^T \times m = 0 \iff m \in (C^\perp)^\perp \iff m \in C$
car $\mathbb{F}_q^n$ est de dimension finie et donc $(C^\perp)^\perp = C$.
Remarquons que toute matrice de contrôle vérifie que la transposée de ses lignes est une base de $C^\perp$.

Proposition 3.2.1
Soit C un code de matrice de contrôle H alors d_C est le nombre minimum de colonnes de H telles qu'elles soient liées.
Preuve
Soit $m = (m_0, ..m_{n-1}) \in C$ et notons $(H_0, ..H_{n-1})$ les colonnes de H alors, comme $m \in C$,
$\sum_{i=0}^{n-1} m_i H_i = 0$, on a que le nombre de colonnes mise en jeu pour la combinaison linéaire est $w(m)$ ce qui conclut.
Remarque: Comme $n - k + 1$ colonnes de $\mathscr{M}_{n-k,1}(\mathbb{F}_q)$ sont forcement liés on obtient une preuve pour la borne de Singleton.

4 Code Cyclique

4.1 Définition et Analogie

Un code cyclique C est un code **linéaire** vérifiant de plus $\forall m = (m_0, ...m_{n-1}) \in C \Rightarrow (m_{n-1}, m_0, ...m_{n-2}) \in C$.

Notons désormais $P_C = \{\sum_{i=0}^{n-1} m_i X^i \in \frac{\mathbb{F}_q[X]}{(X^n-1)}, (m_0, ...m_{n-1}) \in C\}$.
Soit $m(X) \in P_C$, $X \times m(X) = m_{n-1}X^n + \sum_{i=0}^{n-2} m_i X^{i+1} \in P_C$ car $(m_{n-1}, m_0, ...m_{n-2}) \in C$.
Soit $Q \in \frac{\mathbb{F}_q[X]}{(X^n-1)}$ alors $Q := \sum_{i=0}^{n-1} q_i X^i$ et $Q \times m(X) := \sum_{i=0}^{n-1} q_i m(X) X^i \in P_C$ car C est cyclique
C étant linéaire on a donc que P_C est un idéal de $\frac{\mathbb{F}_q[X]}{(X^n-1)}$.[2]

Proposition 4.1.1
Les idéaux de $\frac{\mathbb{F}_q[X]}{(X^n-1)}$ sont principaux et engendrés par un polynôme unitaire $g(X)|(X^n-1)$.
Preuve
Soit I un idéal de $\frac{\mathbb{F}_q[X]}{(X^n-1)}$. Considérons π la projection canonique de $\mathbb{F}_q[X]$ vers $\frac{\mathbb{F}_q[X]}{(X^n-1)}$, on sait que $\pi^{-1}(I)$ est un idéal de $\mathbb{F}_q[X]$ or $\mathbb{F}_q[X]$ est principal, dès lors $\pi^{-1}(I) = g(X) \times \mathbb{F}_q[X]$ et comme I contient 0 on a ainsi $X^n - 1 \in \pi^{-1}(I)$ d'où $g(X)|(X^n-1)$.
De plus comme π est surjective on en déduit $\pi(\pi^{-1}(I)) = I$. Ainsi $I = \overline{g} \times \frac{\mathbb{F}_q[X]}{(X^n-1)}$.

Proposition 4.1.2
Les idéaux de $\frac{\mathbb{F}_q[X]}{(X^n-1)}$ sont exactement ceux engendrés par un polynome $g(X)|(X^n-1)$.
Preuve
On a déjà montré que les idéaux de $\frac{\mathbb{F}_q[X]}{(X^n-1)}$ était de cette forme il suffit donc de vérifier qu'ils conviennent. Soit $g(X)|(X^n-1)$ et considérons $I_g = \overline{g(X)} \times \frac{\mathbb{F}_q[X]}{(X^n-1)}$.
$I_g = \pi(I)$ avec $I = g(X) \times \mathbb{F}_q[X]$, I étant un idéal on en conclut que I_g est un idéal car π est surjective

Proposition 4.1.3
$(\overline{g}, \overline{Xg}, \overline{X^2 g}, ...\overline{X^{n-1-deg(g)}g})$ est une base de P_C=$\{\sum_{i=0}^{n-1} m_i X^i \in \frac{\mathbb{F}_q[X]}{(X^n-1)}, (m_0, ...m_{n-1}) \in C\}$, ainsi $k = n - deg(g)$.[2]
Preuve
La famille est libre : Pour tout $i \in [\![0, n-1-deg(g)]\!]$, $deg(X^i g) \leq n-1 < n$.
Supposons qu'il existe $\lambda_0, ..\lambda_{n-1-deg(g)} \in \mathbb{F}_q$ tel que $\sum_{i=0}^{n-1} \lambda_i \overline{X^i g} = \overline{0}$ on a dès lors $\overline{\sum_{i=0}^{n-1} \lambda_i X^i g} = \overline{0}$ or par ce qui precède on en déduit que $\lambda_0 = ... = \lambda_{n-1-deg(g)} = 0$.
La famille est génératrice : Soit $m \in P_C$ alors $m = \overline{g} \times \overline{h}$, $h \in \mathbb{F}_q[X]$, quitte à operer la division euclidienne par $X^n - 1$ on peut supposer $deg(g \times h) \leq n-1$ d'ou $deg(h) \leq n - 1 - deg(g)$ ce qui permet de conclure.

Proposition 4.1.4
Écrivons $g = \sum_{i=0}^{n-k} a_i X^i$,

Une matrice génératrice de C est G = $\begin{pmatrix} a_0 & 0 & & 0 \\ a_1 & a_0 & & 0 \\ . & a_1 & & . \\ a_{n-k} & . & & 0 \\ & & \vdots & \\ 0 & a_{n-k} & & a_0 \\ . & 0 & & a_1 \\ . & . & & . \\ 0 & 0 & & a_{n-k} \end{pmatrix}$.

Appelons $h = \frac{X^n-1}{g} = \sum_{i=0}^{k} b_i X^i \in \mathbb{F}_q[X]$

Une matrice de contrôle de C est H = $\begin{pmatrix} b_k & ... & b_0 & 0 & ... & 0 & 0 \\ 0 & b_k & ... & b_0 & 0 & ... & 0 \\ & & & \vdots & & & \\ 0 & 0 & ... & 0 & b_k & ... & b_0 \end{pmatrix}$.

Preuve

Comme $(\overline{g}, \overline{Xg}, \overline{X^2g}, ...\overline{X^{n-1-deg(g)}g})$ est une base de P_C on en déduit l'expression d'une matrice génératrice de C.
Soit $m = (m_0, ...m_{n-1}) \in C$, Montrons que $H \times m = 0$:
$\forall i \in [\![0; n-k-1]\!]$ $\sum_{j=0}^{k} m_{j+i} \times b_{k-j} = 0$ car cela correspond au coefficient devant x^{i+k} dans le produit $m(X) \times h(X)$ qui est nul car g divise $m(X)$.
Ainsi les lignes de H sont dans $C^{\perp}$ or les lignes de H forment une famille libre, dès lors il s'agit d'une base de par la dimension de $C^{\perp}$. D'où H est une matrice de contrôle de C.

4.2 Codage

Soit $m = (m_0, ..m_{k-1}) \in \mathbb{F}_q^k$ on considère le polynome associé $m(X) = \sum_{i=0}^{k-1} m_i X^i \in \frac{\mathbb{F}_q[X]}{(X^n-1)}$.
Pour l'encoder on renvoie $m(X) \times g(X) \in P_C$.

4.3 Décodage

Soit $m \in \frac{\mathbb{F}_q[X]}{(X^n-1)}$, pour le décodage on définit le syndrome d'un mot qu'on note $S(m)$ qui est le reste de la division euclidienne de m par g.
Cette fonction est bien définie, $\forall a, b \in \mathbb{F}_q[X]$ on a que le reste de la division euclidienne de $a + b(X^n - 1)$ est le reste de la division euclidienne de a par g car g divise $X^n - 1$.
Proposition 4.3.1

- S est linéaire.
- Soit $m \in \frac{\mathbb{F}_q[X]}{(X^n-1)}$, $S(m) = 0 \iff m \in P_C$.
- $\forall (m, e) \in P_C \times \frac{\mathbb{F}_q[X]}{(X^n-1)}$, $S(m + e) = S(m)$.
- Soit $e, e' \in \frac{\mathbb{F}_q[X]}{(X^n-1)}$ tel que $S(e) = S(e')$. Si $w(e) \leq e_C$ et $w(e') \leq e_C$, alors $e = e'$.

Preuve

S est linéaire car si $a, a' \in \mathbb{F}_q[X]$ tel que $a = q \times g + r$ et $a' = q' \times g + r'$ avec $deg(r) < deg(g)$ et de même pour r' on a $a + a' = (q + q') \times g + (r + r')$ et $deg(r + r') \leq deg(a + a') - 1$. Ainsi on a $S(\overline{a + a'}) = S(\overline{a}) + S(\overline{a'})$
Par définition de P_C on a $m \in P_C \iff g$ divise un représentant de $m \iff S(m) = 0$
Du point 1 et 2 on obtient directement le point 3.
Supposons $e, e' \in \frac{\mathbb{F}_q[X]}{(X^n-1)}$ avec $S(e) = S(e')$ et $w(e) \leq e_C, w(e') \leq e_C$.
Alors par le point 1 et 2 on a $S(e - e') = 0$ d'ou $e - e' \in P_C$ et $w(e - e') \leq w(e) + w(e') \leq d_C - 1$, d'ou par définition de la distance minimale on a donc $e = e'$.

Construisons désormais le décodage[3]:

On initialise dans un premier temps un tableau T qui sera constitué des différents polynômes $S(\overline{m})$ pour $m \in \mathbb{F}_q[X]$ tel que $deg(m) = n - 1$ et m a au plus e_C composantes non nul.

On prend un mot $y = m + e$ avec $m \in C$ et e une erreur telle que $w(e) \leq e_C$ on a dès lors $S(X^i y) = S(X^i e)$ pour $i \in [\![0; n-1]\!]$ car $X^i m \in P_C$. Pour le calcul de $S(X^i y)$ on pourra s'aider du résultat suivant :

Proposition 4.3.2
$S(Xm) = XS(m) - s_{n-k-1} g(X)$ si $S(m) = \sum_{i=0}^{n-k-1} s_i X^i$
Preuve

Tout d'abord $n - k = deg(g)$ et par définition d'une division euclidienne on a $deg(S(m)) \leq n - k - 1$.
Ecrivons $S(m) = \sum_{i=0}^{n-k-2} s_i X^i + s_{n-k-1} X^{n-k-1}$, g étant unitaire on a donc $s_{n-k-1} X^{n-k} = s_{n-k-1}(g - \tilde{g})$.
Avec $\tilde{g}$ de degré inférieur ou égal à $n - k - 1$ et vérifiant $g = X^{n-k} + \tilde{g}$
Ainsi comme $m = q \times g + S(m)$ on a $Xm = Xqg + XS(m) = Xqg + s_{n-k-1}(g - \tilde{g}) + X \sum_{i=0}^{n-k-2} s_i X^i$
avec $\tilde{g}$ de degré inférieur ou égal à $n - k - 1$ et vérifiant $g = X^{n-k} + \tilde{g}$
D'ou $S(Xm) = X \sum_{i=0}^{n-k-2} s_i X^i - s_{n-k-1}\tilde{g} = XS(m) - s_{n-k-1}\tilde{g} - X^{n-k} s_{n-k-1} = XS(m) - s_{n-k-1}(\tilde{g} + X^{n-k}) = XS(m) - s_{n-k-1} g$.

Ainsi on calcule $S(X^i y)$ pour i $\in [\![0; n-1]\!]$ jusqu'à ce qu'on tombe sur une valeur calculée dans le tableau $S(e')$, ce qui sera toujours possible si l'erreur est de poids inférieur à e_C car on augmentera le degré de l'erreur jusqu'à quelle soit de degré $n-1$. Si le syndrome coïncide avec un élément du tableau on en conclut que $X^i e = e'$ car $S(X^i y) = S(X^i e)$ et donc $e = X^{n-i} e'$. Il suffit donc de renvoyer $y - X^{n-i} e'$ pour décoder le message initial.
Remarque : On conserve dans le tableau $T : \sum_{i=0}^{e_C - 1} \binom{n-1}{i} \times (q-1)^{i+1}$ éléments grâce au caractère cyclique de notre code. Sans l'hypothèse de code cyclique on devrait conserver $\sum_{i=0}^{e_C} \binom{n}{i} \times (q-1)^{i+1}$ éléments.

5 Application

Pour l'exemple on prendra $p = 2$ et on pose $g(X) = X^8 + X^7 + X^6 + X^4 + 1$ qui divise $X^{15} - 1$ dans $\mathbb{F}_2$. on a dès lors que le polynôme de contrôle est $h(X) = X^7 + X^6 + X^4 + 1$ et on peut en déduire à l'aide des **propositions 3.2.1/4.1.4** que $d_C = 5$. On obtient ainsi un code de paramètre $(15, 7)$ et de distance minimale 5[3]. Un rapide calcul probabiliste dans le cas où chaque bit a une probabilité de $\frac{1}{30}$ d'être modifié nous permet de conclure que le mot initial sera retrouvé dans 98.75% des cas. Pour l'implémentation des polynômes de $\mathbb{F}_2[X]$ j'ai décidé de les représenter par des entiers, au polynôme $P = \sum_{i=0}^{n-1} p_i X^i \in \mathbb{F}_2[X]$, on associe l'entier $p = \sum_{i=0}^{n-1} p_i 2^i$. Ce qui permet de réaliser des opérations plus efficacement en pratique qu'avec une représentation par liste.
On obtient ainsi le résultat suivant après application de l'algorithme d'encodage et de décodage:

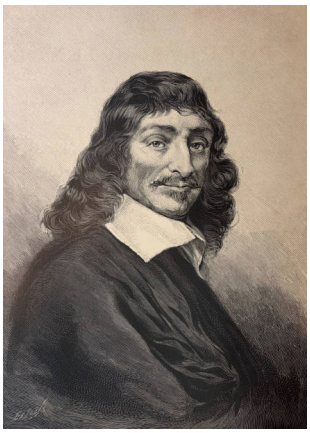
Image initiale


Image déformée


Image transmise

6 Conclusion

Finalement les codes cycliques fournissent un grand nombre de codes correcteurs d'erreurs ainsi qu'un encodage et décodage aisé. L'exemple donne un code cyclique qui corrige une grande partie des erreurs.Néanmoins mon implémentation nécessite environ 25 minutes pour l'encodage et le décodage d'une image HD. De plus les erreurs viennent généralement par paquets et notre code n'est pas adapté pour détecter ce genre d'erreurs.On n'a aussi pas pris en compte la suppression d'une lettre ce qui complexifie énormément le problème.

7 Bibliographie

[1] Pierre Abbrugiati, Introduction aux codes correcteurs d'erreurs, https://www.lirmm.fr/ chaumont/download/cours/codesc
[2] A.A. Pantchichkine, Mathématiques des codes correcteurs d'erreurs
https://www-fourier.ujf-grenoble.fr/ panchish/04cc
[3] Martin Borello, Code correcteurs 1 https://www.math.univ-paris13.fr/ borello/codes1/20202021/codescorrecteurs1.pdf

8 Annexe

```
from PIL import Image
import random
random.seed()
n=15
k=7
e=2
g=465
path = 'images\descartes.jpg'
img = Image.open(path)
pixels = img.load()
image_blank=Image.new('RGB',(img.size[0],img.size[1]))
pixel_sans_code = image_blank.load()

def decalage(p, d):
    d=d%n
    nombre = (p << d)|(p >> (n - d))
    return nombre % (1<<n)

def multiplication(p,q):
    s = 0
    j = len(bin(p))
    for i in range(j-2):
        if(bin(p)[j-i-1]=='1'):
            s = s ^ (q<<i)
    return s

def deg(n):
    return len(bin(n))-3

def division_euclid(a,b): # a = bq+r
    r = a
    q = 0
    while(deg(r)>=deg(b)):
        q=q+2**(deg(r)-deg(b))
        t=b<<(deg(r)-deg(b))
        r=r ^ t
    return (q,r)

def reste(a,b):
    (q,r) = division_euclid(a,b)
    return r

def quotient(a,b):
    (q,r) = division_euclid(a,b)
    return q

def poids(w):
    total = 0
    j = len(bin(w))
    for i in range(j-2):
        if(bin(w)[j-i-1]=='1'):
            total = total+1
    return total

def encodage(m,g):
    return multiplication(m,g)

def calcul(g,s):
    if (deg(s)<n-k-1):
        return decalage(s,1)
    else:
        a = decalage(s,1)
        return a^g

def decodage_naif(w,g):
    (a,s) = division_euclid(w,g)
    if(poids(s)==0):
        return a
    return 0

T=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
for i in range(14):
```

```
    nombre = 2**14 + 2**i
    T[i]=reste(nombre,g)
T[14] = reste(2**14,g)

def decodage(w,g):
    (a,s) = division_euclid(w,g)
    if(s==0):
        return a
    for i in range(n):
        if(s in T):
            k = T.index(s)
            erreur = 2**14+2**k
            if (k == 14):
                erreur = 2**14
            erreur = decalage(erreur,n-i)
            m = w^erreur
            return quotient(m,g)
        else:
            s=calcul(g,s)
    return a

def ajout_erreur(w):
    p=w
    for k in range(n):
        if(random.randint(0,15)==0):
            p = p ^ (1<<k)
    return p

for i in range(img.size[0]):
    for j in range(img.size[1]):
            print(nombre)
            (p1,p2,p3)=pixels[i,j]
            p1=p1>>1
            p2=p2>>1
            p3=p3>>1
            p1=encodage(p1,g)
            p2=encodage(p2,g)
            p3=encodage(p3,g)
            p1=ajout_erreur(p1)
            p2=ajout_erreur(p2)
            p3=ajout_erreur(p3)
            pixel_sans_code[i,j]=(2*decodage_naif(p1,g),2*decodage_naif(p2,g),2*
    decodage_naif(p3,g))
            pixels[i,j]=(2*decodage(p1,g),2*decodage(p2,g),2*decodage(p3,g))

img.save(r'test\ga_code.png')
image_blank.save(r'test\ga_sans_code.png')
```