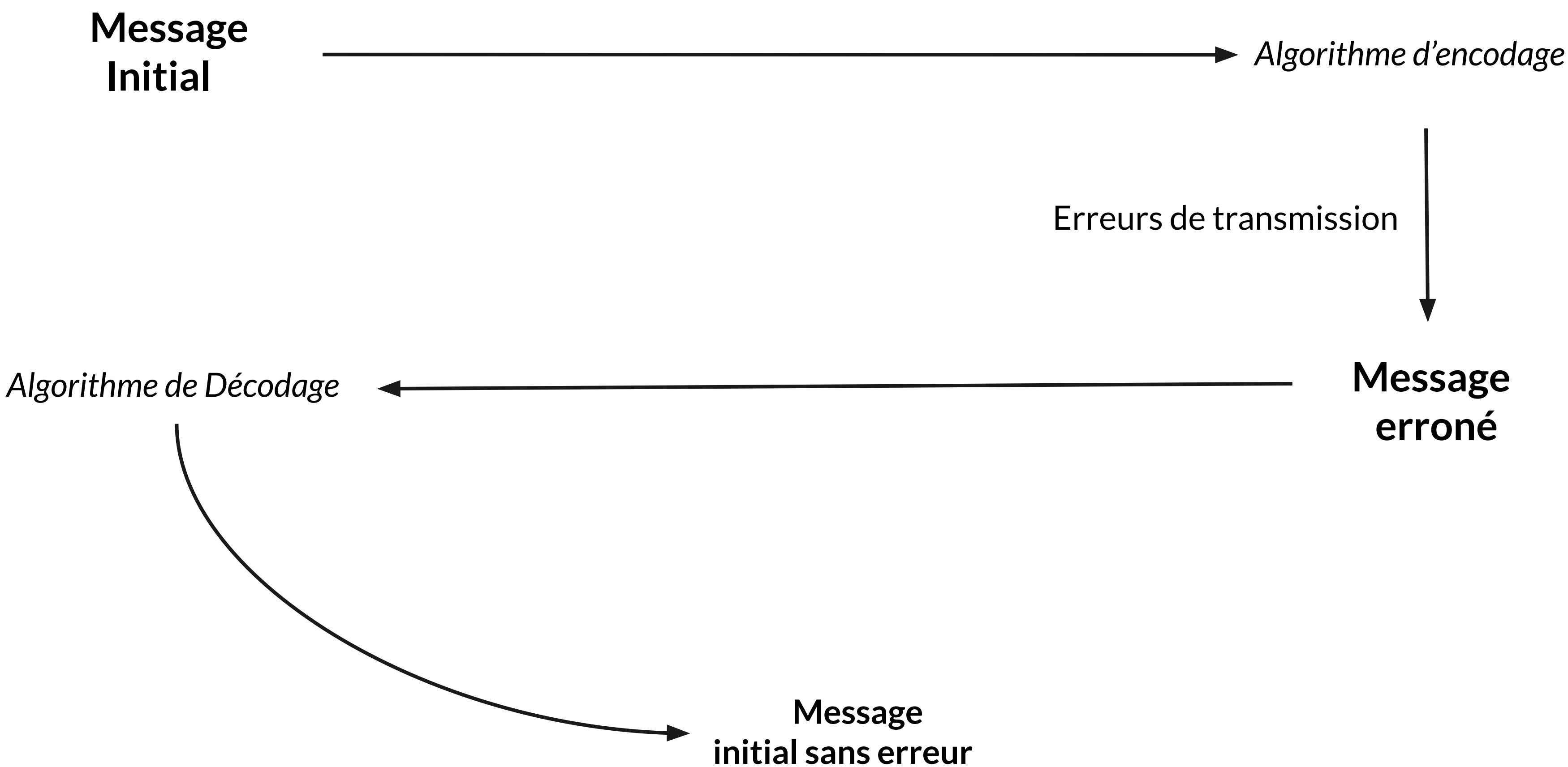
Codes Cycliques

Tarik Ouadjou

35114

2022/2033

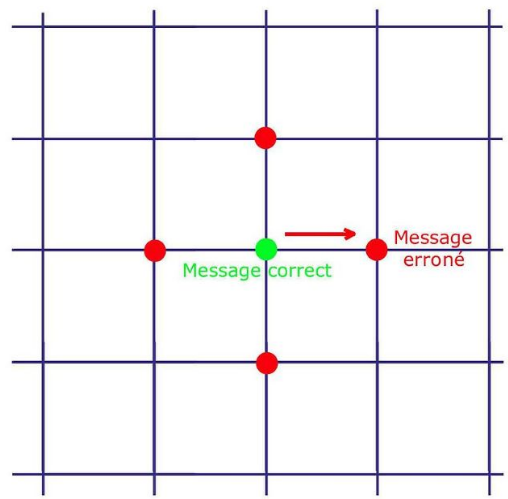
Contexte


Message
Initial
Algorithme d'encodage
Erreurs de transmission
Message
erroné
Algorithme de Décodage
Message
initial sans erreur

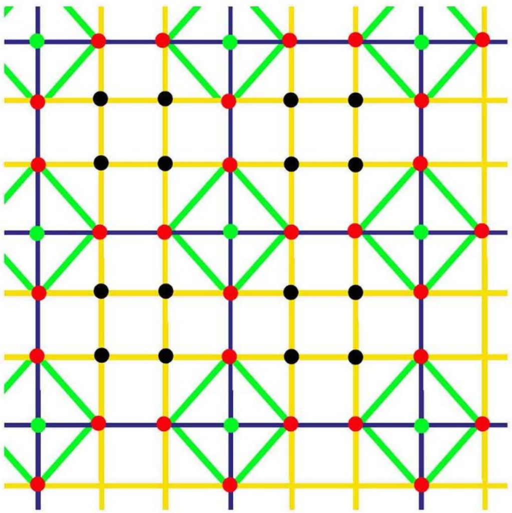
Exemple de l'alphabet phonétique

On considère le message : **TIPE**

Encoder :



Code sans redondance



Code correcteur

https://fr.wikipedia.org/wiki/Code_correcteur

Correction :

On renvoie le mot TIPE

Mise en situation

Communication satellites :

- Téléphonie
- Télévision / Radio
- Internet : Starlink
- GPS

Problèmes :

- Interférences
- Distance

https://www.journaldunet.com/economie/services/1424539-les-satellites-nouvelle-source-de-donnees-pour-les-smart-cities/

Problématique : Comment corriger les erreurs de transmission de façon efficace ?

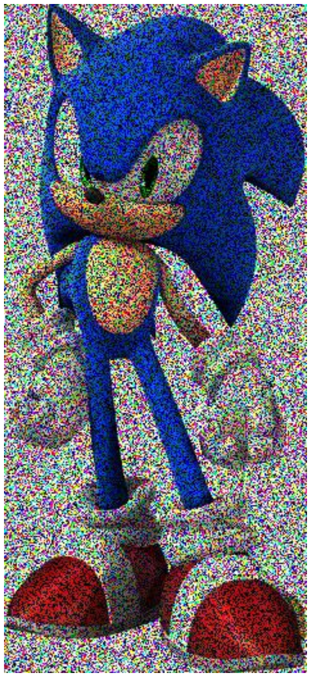
Cahier des charges

Modèle :

Probabilité d’erreur : une lettre sur 30 est modifiée
On travaille sur les **écritures binaires**

On cherche une structure qui vérifie :

- Conserve 99% des données en moyenne
- Impact la mémoire d’au plus un facteur 2
- Décodage efficace, pouvoir décoder une image de taille standard en un temps raisonnable



https://www.123-stickers.com/autocollant-sonic/4685-sticker-sonic.html

Définitions élémentaires

Lettre : Un élément de $\mathbb{F}_2$

Mot : Un mot de taille n est un élément de $\mathbb{F}_2^n$

Distance de Hamming :

d(x,y) est le nombre de lettres qui diffèrent entre les mots x et y

Propriété :
La distance de Hamming est une distance

Distance minimale :

$$dC = \min\{d(x,y) | (x,y) \in C^2 \ x \neq y\}$$

Poids d'un mots :

Nombre de composantes non nulles du mot, on note cette fonction ***w***

Définitions

Un **Code Linéaire C** de paramètres (n,k) est un sous-espace vectoriel de $\mathbb{F}_2^n$ de dimension k

Fonction d'encodage φ linéaire injective

$$\text{Mot de } \mathbb{F}_2^k \xrightarrow{\hspace{6cm}} \text{Mot de } \mathbb{F}_2^n$$

Remarque: $dC = \min\{w(m) | m \in C \; m \neq 0\}$

Un **Code Cyclique C** est un code linéaire vérifiant de plus :

$$\forall m = (m_0, ...m_{n-1}) \in C \implies (m_{n-1}, m_0...m_{n-2}) \in C$$

Exemple :

C = { 000, 011 , 101 , 110 } est un code cyclique

Analogie entre $\mathbb{F}_2^n$ et $\mathbb{F}_2[X]/(X^n - 1)$

$$\forall m = (m_0, ...m_{n-1}) \in C,\ m(X) = \sum_{i=0}^{n-1} m_i X^i$$

<u>Stabilité de C</u> :

$$X \times m(X) = m_{n-1}X^n + \sum_{i=0}^{n-1} m_i X^i := (m_{n-1}, m_0, ..m_{n-2})$$

Ainsi C est stable par produit par tout polynôme

<u>Propriété</u> :

C est un **idéal** de $\mathbb{F}_2[X]/(X^n - 1)$

Polynôme minimal

On appelle **polynôme minimal** **g** de C l'unique polynôme tel que :

$$C = \{g(X) * h(X) |\ h \in \mathbb{F}_2[X]/(X^n - 1)\}$$

Propriété :

$(g(X), Xg(X), X^2 g(X), ..., X^{n-1-deg(g)} g(X))$ est une base de C

Ainsi k = n-deg(g)

Théorème :

On a $g | X^n - 1$ et réciproquement tout diviseur g de $X^n - 1$ engendre un code cyclique de paramètre (n,n-deg(g))

Encodage

On <u>fixe</u> :

- n = 15
- g = $X^8 + X^7 + X^6 + X^4 + 1$ $\quad g|(X^{15} - 1)$
- k = 7

Proposition :
La distance minimale du code engendré par g est 5

<u>**Encodage**</u> :

Soit $m = (m_0, ..m_{k-1}) \in \mathbb{F}^k$ l'encodage de m est : $m(X) * g(X) = (\sum_{i=0}^{k-1} m_i X^i) * g(X)$

Syndrome

Définition :

On définit le Syndrome d'un mot m $S(m)$ qui est le reste de la division euclidienne de m par g pour un mot m

Remarque:

La fonction S est bien définie :

$\forall a, v \in \mathbb{F}_2[X] S(a(X^n - 1) + v) = S(v)$ car g(X) | $X^n - 1$

Proposition :

$S(Xm) = XS(m)$ si deg(m) < n-k-1

$S(Xm) = XS(m) - g(X)$ si deg(m) = n-k-1

Propriétés

Proposition:

La fonction S vérifie les propriétés suivantes :

- S est linéaire
- S(m) = 0 si et seulement si $m \in C$
- $\forall m \in C \; S(m+e) = S(e)$
- Si $w(e) < \lfloor \frac{d-1}{2} \rfloor$ alors $S(e) = S(e') \implies e = e'$

Preuve:

Supposons S(e) = S(e') ainsi $e - e' \in C$

$$w(e-e') \leq w(e) + w(e') \leq dC - 1 \implies e - e' = 0$$

Décodage de Meggitt

<u>**Algorithme de décodage :**</u>

<u>*Etape 1*</u> :

Calcul de : S(e(X)) avec e de poids 2 et $e_{n-1} \neq 0$ et les introduits dans un tableau

<u>Etape 2</u> :

Posons y = m + e avec $m \in C$, si S(y) = 0 alors $y \in C$ sinon on a S(y) = S(e)

<u>Etape 3</u> :

Si S(y) appartient au tableau calculé alors on a trouvé l'erreur e(x) et on renvoie y-e

<u>Etape 4</u> :

On calcul $S(X^i y)$ jusqu'à avoir un élément du tableau, on renvoie alors $y - x^{n-i}e$

Proposition :

S'il y a une erreur de poids inférieur à 2 l'algorithme la détecte et la corrige

Exemple

Table des syndromes:

e(X)	S(e(X))	e(X)	S(e(X))	e(X)	S(e(X))
X^{14}	$X^7+X^6+X^5+X^2$	$X^{14}+X^9$	$X^7+X^4+X^3+X+1$	$X^{14}+X^4$	X^7+X^6
$X^{14}+X^{13}$	$X^7+X^4+X^3+X^2$	$X^{14}+X^8$	$X^5+X^4+X^3+1$	$X^{14}+X^3$	X^7+X^6
$X^{14}+X^{12}$	$X^7+X^6+X^4+X$	$X^{14}+X^7$	$X^6+X^5+X^3$	$X^{14}+X^2$	X^7+X^6
$X^{14}+X^{11}$	$X^7+X^6+X^5+X^4+X^2+1$	$X^{14}+X^6$	$X^7+X^5+X^3$	$X^{14}+X$	X^7+X^6
$X^{14}+X^{10}$	X^3+X^2+X	$X^{14}+X^5$	$X^7+X^6+X^3$	$X^{14}+1$	X^7+X^6

Exemple:

Soit m(X) = $1+X^2+X^4$ l'encodage de m est y(X) = $1+X^2+X^7+X^8+X^9+X^{11}+X^{12}$

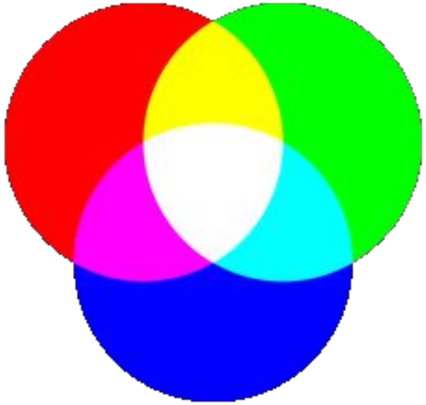
introduisons l'erreur e(X) = X^3+X^{12} ainsi y(X) = $1+X^2+X^3+X^7+X^8+X^9+X^{11}$

S(y) = X^5+X^4+X, S(Xy) = XS(y) = $X^6+X^5+X^2$, $S(X^2y) = XS(Xy) = X^7+X^6+X^3$

On trouve l'erreur $e(X) = X^{15-2}(X^{14}+X^5) = X^{12}+X^3$

Implémentation par liste

Un **pixel** est un triplet de nombre entre 0 et 255 (p1,p2,p3)



https://fr.science-questions.org/comment_ca_marche/162/Les_pixels_de_la_television_en_couleur/

$$p = \sum_{i=0}^{7} a_i 2^i \longrightarrow p = (a_7, ...a_1) \in \mathbb{F}_2^k$$

On représente le polynôme associé au message transmis par une liste

Résultats

Image initiale

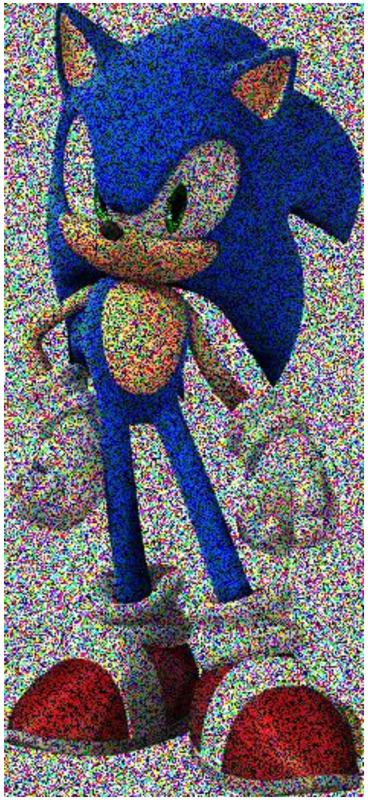


Image déformée

Image corrigée

Implémentation par bit

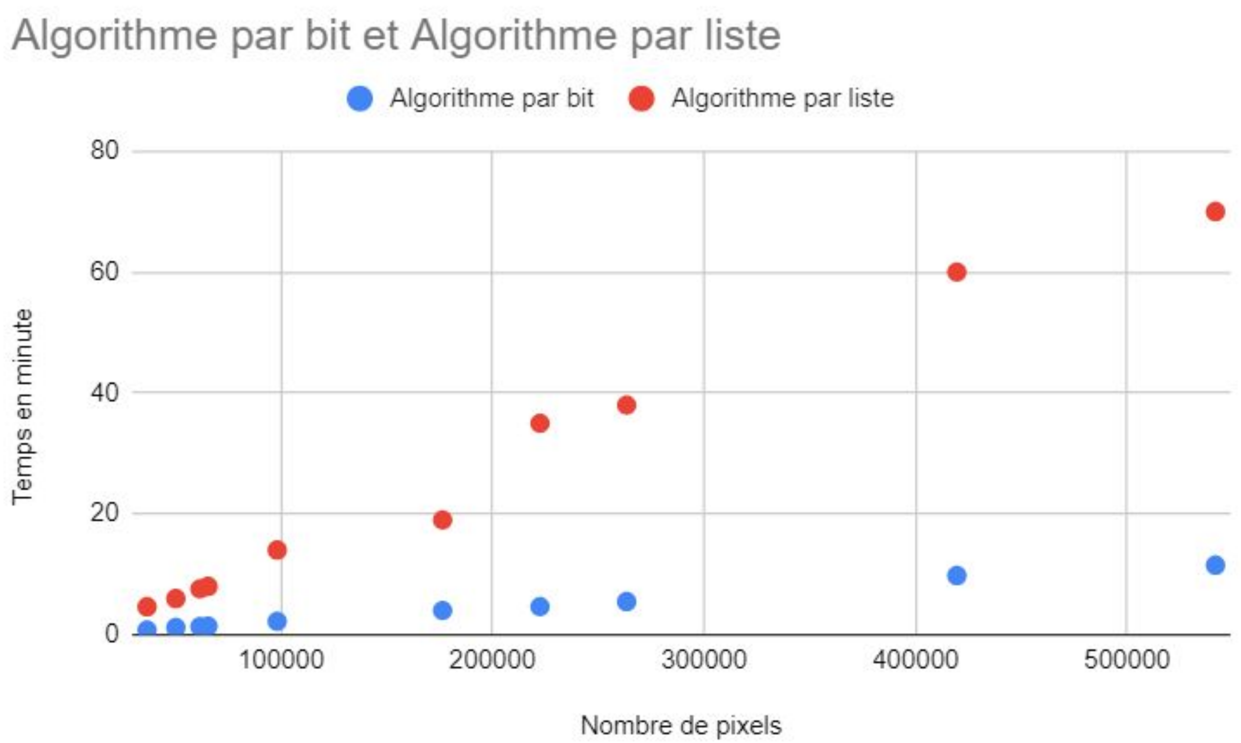
$$\sum_{i=0}^{14} a_i X^i \longrightarrow p = \sum_{i=0}^{14} a_i 2^i$$

On travail sur les polynômes de manière implicite :

- L'addition de 2 polynômes : Xor des 2 entiers associés
- Multiplication par X :

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

Complexité



Représentation par liste

Coefficient directeur : 8000 Pixels par minute

Représentation par bit

Coefficient directeur : 45700 Pixels par minute

L'algorithme est <u>**6**</u> fois plus efficace qu'initialement

Analyse probabiliste

<u>Nombre erreurs</u>:

On appelle nombre d'erreurs X la variable aléatoire qui compte le nombre d'erreurs d'un mot m,

Proposition :

Le nombre d'erreurs d'un mot m suit une loi de Bernoulli de paramètre (n,p) lorsque les erreurs sont introduites selon une loi uniforme sur chacun des bits

Conséquence :

L'algorithme est capable de retrouver le message initial dans au moins 98.75% des cas

Conclusion

Respect des conditions imposées :

- L'algorithme corrige une grande partie des erreurs
- Nécessite deux fois plus de mémoire
- Inutilisable en pratique : environ 25 minutes pour décoder une image en HD (1280 * 720 pixels)

Modèle simpliste :

- Erreurs viennent par blocs
- Suppression de bits

Annexes

```python
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)
```

```python
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)
```

Annexes

```
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(0,29)==0):
            p = p ^ (1<<k)
    return p
```