IFT 6170 Théorie et pratique des codes correcteurs

Claude Crépeau

Département d'Informatique et R.O.,
Université de Montréal,
C.P. 6128, succursale centre-ville,
Montréal (Québec), Canada H3C 3J7.
e-mail: crepeau@iro.umontreal.ca.

Table of Contents

1 Introduction

La théorie des codes correcteurs étudie les façons de transmettre, sur un canal imparfait, des messages avec grande fiabilité. Soit A un ensemble de M mots à transmettre au moyen d'un canal imparfait. Un code C associé à A est un ensemble de M mots, nommés *mots de code*, sur un ensemble B. Dans la plupart des cas A sera l'ensemble des mots de k bits, $A = \{0,1\}^k$, et B sera l'ensemble des mots de n bits, $B = \{0,1\}^n$, mais dans certains cas il en sera autrement. On dénote les caratéristiques d'un code de M mots sur $\{0,1\}^n$ au moyen des trois paramètres (n, M, d) où d est la distance minimale entre deux mots de C au sens de Hamming, $d = \min\{\Delta(c_1, c_2) : c_1, c_2 \in C\}$. La distance de Hamming entre deux mots c_1, c_2, dénotée $\Delta(c_1, c_2)$, est le nombre de positions où c_1 et c_2 diffèrent. En général, un code est défini au moyen d'une application d'*encodage* ϕ de A dans B, ayant C pour image. La fonction ϕ^{-1} de C dans A inverse de la fonction d'encodage ϕ, est dite de *décodage*.

Exemple Considérons le code suivant sur $\{0,1\}^6$ associé à l'ensemble $\{0,1\}^2$:

$$C = \{000000, 010101, 101010, 111111\}$$

Il s'agit d'un code $(6, 4, 3)$, puisque $d = 3$ comme le montre cette liste:

$$\Delta(000000, 010101) = 3$$
$$\Delta(000000, 101010) = 3$$
$$\Delta(000000, 111111) = 6$$
$$\Delta(010101, 101010) = 6$$
$$\Delta(010101, 111111) = 3$$
$$\Delta(101010, 111111) = 3.$$

Dans ce cas on peut définir les fonctions d'encodage $\phi : \{0,1\}^2 \to \{0,1\}^6$ et de décodage $\phi^{-1} : C \to \{0,1\}^2$ par

$$\phi(x) = xxx,$$
$$\phi^{-1}(xxx) = x.$$

Le sens des fonctions d'encodage et de décodage est que lorsqu'un mot x de A doit être transmis, le mot $\phi(x)$ sera transmis à sa place. Quand un mot c de C est reçu, on lui associe le mot $\phi^{-1}(c)$ de A. L'avantage de transmettre c plutot que x est que tous les mots de A sont des mots possibles et donc si x est reçu de façon erronée, un autre mot correct sera alors reçu. Au contraire, si c est reçu de façon erronée, il est probable que le mot ne sera pas dans C et sera alors identifié comme une erreur.

On nomme *détection d'erreurs* le processus par lequel on identifie qu'un mot reçu n'appartient pas à l'ensemble C. Dans l'exemple ci-haut, on détecte une erreur si un mot reçu n'a pas la forme xxx. On nomme enfin *correction d'erreurs* le processus par lequel le mot de code *le plus probable* est substitué à un mot reçu de façon erronée. On entend par "plus probable" le mot de code ayant la plus grande probabilité d'avoir produit le mot erroné reçu. Cette valeur dépendra en général du modèle d'erreur considéré. Dans la plupart des cas qui nous intéressent, le mot de code le plus probable est le plus proche au sens de Hamming.

1.1 Objectifs

Les objectifs premier de la théorie des codes est de trouver des constructions efficaces permettant d'obtenir des codes pour lesquels on sais réaliser de façon efficace les opération suivantes:

- encodage
- décodage
- détection d'erreurs
- correction d'erreurs.

De plus, la théorie des codes s'intéresse à des considération économiques relatives au taux d'expansion des mots transmis (k/n) par rapport au nombres d'erreurs qui peuvent être effifacement détectées et corrigées.

1.2 Codes linéaires

La plupart des codes que nous étudirons, sont de codes *linéaires*. Un code binaire est linéaire s'il satisfait la propriété de linéarité
$$c_1, c_2 \in C \Rightarrow c_1 \oplus c_2 \in C.$$
Il est à noter qu'il en découle qu'un code linéaire sur $\{0,1\}^n$ contient toujours le mot 0^n. De plus, la distance minimale du code est donnée par l'expression suivante $d = \min\{w(c) : c \in C - \{0^n\}\}$ où $w(c)$ est le poids de c, c'est-à-dire le nombre de positions non-nulles de c.

Le code de l'exemple ci-haut est linéaire car

$$000000 \oplus 010101 = 101010 \oplus 111111 = 010101$$
$$000000 \oplus 101010 = 010101 \oplus 111111 = 101010$$
$$000000 \oplus 111111 = 010101 \oplus 101010 = 111111.$$

2 théorie des nombres et corps finis

2.1 définitions

$$
\begin{aligned}
a|b &\iff \exists k \in Z\ [b = ak] \\
a \equiv b \pmod{n} &\iff n|(a-b) \\
b \bmod n &\iff \min\{a > 0 : a \equiv b \pmod{n}\} \\
b \operatorname{div} n &\iff \frac{b - (b \bmod n)}{n} \\
g = pgcd(a,b) &\iff g|a, g|b \text{ et } [g'|a, g'|b \Rightarrow g'|g]
\end{aligned}
$$

2.2 opérations de base efficaces

Pour les opérations de base $+, -, \times, \mathrm{mod}, \mathrm{div}$ il suffit d'utiliser les algorithmes *ordinaires* appris à l'école primaire en utilisant la règle suivante pour ne pas travailler inutilement:

$$
a \left\{ \begin{array}{c} + \\ - \\ \times \end{array} \right\} b \bmod n = \left((a \bmod n) \left\{ \begin{array}{c} + \\ - \\ \times \end{array} \right\} (b \bmod n) \right) \bmod n
$$

Les algorithmes ordinaires sont décrit en détail dans Knuth (Vol 2). Pour les nombres très grands on peut utiliser des méthodes plus rapide pour $\times$. Pour ceci consulter le livre de Brassard-Bratley.

2.3 calcul d'une exponentiation rapide

L'idée de cet algorithme est de préserver à chaque itération la valeur de l'expression $xa^e \bmod n$ tout en réduisant la valeur courante de e.

Algorithme 2.1 ($a^e \bmod n$)

- $x \leftarrow 1$,
- **WHILE** $e > 0$ **DO**
- **IF** e est impair **THEN** $x \leftarrow ax \bmod n$,
- $a \leftarrow a^2 \bmod n$, $e \leftarrow e \operatorname{div} 2$,
- **ENDWHILE**
- **RETURN** x.

2.4 calcul du PGCD et d'inverses

L'idée de cet algorithme est de préserver à chaque itération les deux relations $g = ax + by$ et $g' = ax' + by'$tout en réduisant la valeur courante de g. À la fin de l'algorithme, la valeur de g est le $pgcd(a, b)$. Du même coup la valeur finale de x est telle que $ax \equiv g \pmod{b}$. Quand $g = 1$, on en déduit que x est l'inverse multiplicatif de a modulo b.

Algorithme 2.2 (**Euclide** $pgcd(a, b)$)

- $g \leftarrow a$, $g' \leftarrow b$, $x \leftarrow 1$, $y \leftarrow 0$, $x' \leftarrow 0$, $y' \leftarrow 1$,
- **WHILE** $g' > 0$ **DO**
- $k \leftarrow g \operatorname{div} g'$,
- $(\hat{g}, \hat{x}, \hat{y}) \leftarrow (g, x, y) - k(g', x', y')$,
- $(g, x, y) \leftarrow (g', x', y')$,
- $(g', x', y') \leftarrow (\hat{g}, \hat{x}, \hat{y})$,
- **ENDWHILE**
- **RETURN** (g, x, y).

2.5 Nombres premiers

Si l'on désire vérifier la primalité d'un nombre n il est possible d'utiliserl'algorithme probabiliste de Miller-Rabin. Celui-ci repose sur la notion de nombres "pseudo-premier" base a.

Algorithme 2.3 ($pseudo(a, n)$)

1: IF $pgcd(a, n) \neq 1$ **THEN RETURN** "composé",
2: Soient t un nombre impair et s un entier positif tels que $n - 1 = t2^s$
3: $x \leftarrow a^t \bmod n$, $y \leftarrow 1$,
4: FOR $i \leftarrow 1$ **TO** s
5: IF $x = 1$ **AND** $y \neq \pm 1$ **THEN RETURN** "composé",
6: $y \leftarrow x$, $x \leftarrow x^2 \bmod n$,
7: ENDFOR
8: RETURN "pseudo".

Rabin a montré que si n est premier, alors $pseudo(a, n)$ retourne "pseudo" pour tous les a, $0 < a < n$. Il a aussi montré que si n est composé, alors $pseudo(a, n)$ retourne "composé" pour au moins $3n/4$ valeurs de a, $0 < a < n$.

Algorithme 2.4 (**Miller-Rabin** $prem(n, k)$)

1: FOR $i \leftarrow 1$ **TO** k
2: choisir un élément a au hasard, $0 < a < n$,
3: IF $pseudo(a, n) =$ "composé" **THEN RETURN** "composé",
4: ENDFOR
5: RETURN "premier".

On en déduit facilement que si n est premier, alors $prem(n, k)$ retourne toujours "premier" et que si n est composé, alors $prem(n, k)$ retourne "composé" avec probabilité au moins $1 - (1/4)^k$. Donc quand l'algo $prem$ retourne "composé", il dit toujours la vérité. Par contre, quand il retourne "premier" il reste une très faible probabilité qu'il se soit trompé.

Si l'on veut obtenir un nombre premier au hasard, on utilise le théorèmes suivant pour estimer le nombre de fois moyen qu'il nous faudra utiliser l'algorithme 2.4. Soit $\pi(n) = \#\{a : 0 < a \leq n \text{ et } a \text{ est premier}\}$.

Theorem 1. $\lim_{n\to\infty} \frac{\pi(n) \log n}{n} = 1$

2.6 corps premiers

Soit p un nombre premier. Les nombres $0, 1, 2, ..., p - 1$ munis des opérations $+ \bmod p$ et $\times \bmod p$ forment un coprs de p éléments dénoté $\mathcal{F}_p$.

- contient un élément neutre additif (0)
- chaque élément e possède un inverse additif $-e$
- contient un élément neutre multiplicatif (1)
- chaque élément non-nul e possède un inverse multiplicatif e^{-1}
- associativité
- commutativité
- distributivité

Exemples $\mathcal{F}_2 = (\{0, 1\}, \oplus, \wedge)$. $\mathcal{F}_5 = (\{0, 1, 2, 3, 4\}, +, \times)$ définis par

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

2.7 polynômes sur un corps

Un polynômes sur $\mathcal{F}_p$ est spécifié par une suite finie $(a_n, a_{n-1}, ..., a_1, a_0)$ d'éléments de $\mathcal{F}_p$, avec $a_n \neq 0$. Le nombre n est le degré du polynôme. On a sur les polynômes des opérations $+, -, \times$ analogues à celles sur les entiers.

Les sommes et différences se font composantes par composantes en utilisant alors le $+$ et le $-$ du corps considéré.

Les produits se font en sommant tous les produits croisés donnant un certain exposant. Exemple:

$$\begin{aligned}(x^4+x+1)\times(x^3+x^2+x) &= x^4\times(x^3+x^2+x)+x\times(x^3+x^2+x)+1\times(x^3+x^2+x)\\ &= (x^7+x^6+x^5)+(x^4+x^3+x^2+)+(x^3+x^2+x)\\ &= x^7+x^6+x^5+x^4+(1+1)x^3+(1+1)x^2+x\\ &= x^7+x^6+x^5+x^4+x\end{aligned}$$

De même les opérations $g(x)$ mod $h(x)$ et $g(x)$ div $h(x)$ sont définies comme les polynômes $r(x)$ et $q(x)$ tels que $g(x) = q(x)h(x) + r(x)$ avec $deg(r) < deg(h)$. On les obtient en faisant la division formelle de $g(x)$ par $h(x)$ semblable à celle sur les entiers. Exemple:

$$\begin{aligned}x^7+x^6+x^5+x^4+x &= (x^2)\times(x^5+x^2+1)+(x^6+x^5+x^2+x)\\ &= (x^2+x)\times(x^5+x^2+1)+(x^5+x^3+x^2)\\ &= (x^2+x+1)\times(x^5+x^2+1)+(x^3+1)\end{aligned}$$

donc

$$\begin{aligned}(x^7+x^6+x^5+x^4+x) \bmod (x^5+x^2+1) &= x^3+1\\ (x^7+x^6+x^5+x^4+x) \text{ div } (x^5+x^2+1) &= x^2+x+1\end{aligned}$$

Les exponentiations pour des puissances entières modulo un certain polynôme peuvent s'effectuer au moyen de l'algorithme 2.1 et les *pgcd* de deux polynomes ou les inverses multiplicatifs peuvent être calculés grâce à l'algorithme 2.2.

2.8 polynômes irréductibles

Un polynôme $g(x)$ est *irréductible* s'il ne peut s'écrire comme le produit de deux polynômes $h(x), k(x)$ de degrés inférieurs. On utilise le théorème suivant pour trouver de tels polynômes.

Theorem 2. *Soient $l_1, l_2, ..., l_k$ les facteurs premiers de n et $m_i = n/l_i$ pour $1 \leq i \leq k$. Un polynôme $g(x)$ de degré n est* irréductible *sur $\mathcal{F}_p$ si et seulement si*

- $g(x) | x^{p^n} - x$
- *$pgcd(g(x), x^{p^{m_i}} - x) = 1$ pour $1 \leq i \leq k$*

Algorithme 2.5 (Rabin $Irr(p,n)$ **)**

1: soient $l_1, l_2, ..., l_k$ les facteurs premiers de n et $m_i = n/l_i$ pour $1 \leq i \leq k$,
2: REPEAT
3: choisir un polynôme $h(x)$ de degré $n-1$ sur $\mathcal{F}_p$ au hasard, $g(x) \leftarrow x^n + h(x)$,
4: UNTIL $x^{p^n} \bmod g(x) = x$ et $pgcd(g(x), x^{p^{m_i}} - x) = 1$ pour $1 \leq i \leq k$,
5: RETURN g.

On utilise le théorème suivant pour estimer le nombre d'itérations moyen de l'algorithme 2.5.

Theorem 3. *Soit $m(n)$ le nombre de polynômes irréductibles $g(x)$ de degré n de la forme $g(x) = x^n + h(x)$ où $h(x)$ est un polynôme de degré $n-1$. On a*

$$\frac{p^n}{2n} \leq \frac{p^n - p^{n/2}\log n}{n} \leq m(n) \leq \frac{p^n}{n}.$$

$\mathcal{F}_2$		$\mathcal{F}_3$	$\mathcal{F}_5$	$\mathcal{F}_7$
$x+1$	x^9+x^4+1	$x+1$	$x+1$	$x+1$
x^2+x+1	$x^{10}+x^3+1$	x^2+x+2	x^2+x+2	x^2+x+3
x^3+x+1	$x^{11}+x^2+1$	x^3+2x+1	x^3+3x+2	x^3+3x+2
x^4+x+1	$x^{12}+x^6+x^4+x+1$	x^4+x+2	x^4+x^2+x+2	
x^5+x^2+1	$x^{13}+x^4+x^3+x+1$	x^5+2x+1		
x^6+x+1	$x^{14}+x^{10}+x^6+x+1$	x^6+x+2		
x^7+x^3+1	$x^{15}+x+1$			
$x^8+x^4+x^3+x^2+1$	$x^{16}+x^{12}+x^3+x+1$			

Fig. 1. polynômes irréductibles (primitifs) sur $\mathcal{F}_2$ et irréductibles sur $\mathcal{F}_3$,$\mathcal{F}_5$,$\mathcal{F}_7$..

2.9 Corps quelconques

Soit p un nombre premier et n un entier positif. On construit le corps à p^n éléments à partir du corps à p éléments.

- Les éléments de $\mathcal{F}_{p^n}$ ont la forme $e_1 e_2 ... e_n$ où e_i est élément de $\mathcal{F}_p$.
- La somme de deux éléments de $\mathcal{F}_{p^n}$ est définie par

$$e_1 e_2 ... e_n + e'_1 e'_2 ... e'_n = \hat{e}_1 \hat{e}_2 ... \hat{e}_n$$

  tels que $\hat{e}_i = e_i + e'_i$ pour $1 \leq i \leq n$.
- Le produit de deux éléments de $\mathcal{F}_{p^n}$ est défini par

$$e_1 e_2 ... e_n \times e'_1 e'_2 ... e'_n = \hat{e}_1 \hat{e}_2 ... \hat{e}_n$$

  tels que

$$(\hat{e}_1 x^{n-1} + \hat{e}_2 x^{n-2} + ... + \hat{e}_n) = (e_1 x^{n-1} + e_2 x^{n-2} + ... + e_n) \times (e'_1 x^{n-1} + e'_2 x^{n-2} + ... + e'_n) \bmod r(x)$$

  avec $r(x)$ un polynôme de degré n irréductible sur $\mathcal{F}_p$.

Exemples calculs sur $\mathcal{F}_{2^5}$
$10011 + 01110 = (1+0)(0+1)(0+1)(1+1)(1+0) = 11101$

$10011 \times 01110 = 01001$ car $(x^4+x+1) \times (x^3+x^2+x) \bmod (x^5+x^2+1) = x^3+1$.

+	000	001	010	011	100	101	110	111
000	000	001	010	011	100	101	110	111
001	001	000	011	010	101	100	111	110
010	010	011	000	001	110	111	100	101
011	011	010	001	000	111	110	101	100
100	100	101	110	111	000	001	010	011
101	101	100	111	110	001	000	011	010
110	110	111	100	101	010	011	000	001
111	111	110	101	100	011	010	001	000

$\times$	000	001	010	011	100	101	110	111
000	000	000	000	000	000	000	000	000
001	000	001	010	011	100	101	110	111
010	000	010	100	110	011	001	111	101
011	000	011	110	101	111	100	001	010
100	000	100	011	111	110	010	101	001
101	000	101	001	100	010	111	011	110
110	000	110	111	001	101	011	010	100
111	000	111	101	010	001	110	100	011

Fig. 2. opérations de $\mathcal{F}_{2^3}$

2.10 Éléments primitifs

Il existe sur tout corps fini $\mathcal{F}_q$ un *élément primitif*, c'est-à-dire un élément g tel que $g^1, g^2, ..., g^{q-1}$ énumère les $q-1$ éléments non-nuls du corps. On utilise le théorème suivant pour trouver un élément primitif sur $\mathcal{F}_q$.

Theorem 4. *Soient $l_1, l_2, ..., l_k$ les facteurs premiers de $q-1$ et $m_i = (q-1)/l_i$ pour $1 \le i \le k$. Un élément g est* primitif *sur $\mathcal{F}_q$ si et seulement si*

- $g^{q-1} = 1$
- $g^{m_i} \neq 1$ *pour* $1 \le i \le k$

Algorithme 2.6 ($Prim(q)$)

soient $l_1, l_2, ..., l_k$ les facteurs premiers de $q-1$ et $m_i = \frac{q-1}{l_i}$ pour $1 \le i \le k$,
REPEAT
choisir un élément non-nul g de $\mathcal{F}_q$ au hasard,
UNTIL $g^{m_i} \neq 1$ pour $1 \le i \le k$,
RETURN g.

On utilise les théorèmes suivant pour estimer le nombre d'itérations moyen de l'algorithme 2.6. Soit $\phi(n) = \#\{a : 0 < a < n \text{ et } pgcd(a,n) = 1\}$.

Theorem 5. $\#\{g : g$ *est un élément primitif de* $\mathcal{F}_q\} = \phi(q-1)$.

Theorem 6. $\liminf_{n\to\infty} \dfrac{\phi(n)\log\log n}{n} = e^{-\gamma}$

Exemples: 2 est un élément primitif sur $\mathcal{F}_5$ car $\{2, 2^2, 2^3, 2^4\} = \{2, 4, 3, 1\}$.
010 est un élément primitif sur $\mathcal{F}_{2^3}$ comme le montre ce tableau:

x	$-\infty$	0	1	2	3	4	5	6
$(010)^x$	000	001	010	100	011	110	111	101

2.11 Factorisation de $q-1$...

En général, il s'avère difficile de factoriser le nombre $q-1$. Il ne sera donc possible de trouver un élément primitif que pour les nombres $q-1$ dont les facteurs premiers sont connus. Une alternative existe néanmoins si l'on peut se contenter d'un corps contenant un nombre aléatoire d'éléments. Éric Bach [] a développé un algorithme probabiliste efficace capable de générer un entier aléatoire d'une certaine longueur avec sa factorisation. Supposons que l'on obtienne aléatoirement le nombre r avec sa factorisation grâce à l'algortihme de Bach. Il sera alors possible de tester si le nombre $r+1$ est premier ou de la forme p^n pour p premier. Dans ce cas un corps fini de $r+1$ éléments pourra être obtenu et un élément primitif pourra être découvert.

3 Espaces vectoriels

Un espace vectoriel est constitué d'un ensemble de vecteurs V muni d'une addition $\oplus$ ayant une structure de groupe abélien $G = (V, \oplus)$ et d'un ensemble de scalaires S muni d'une addition $+$ et d'un produit $\times$ ayant une structure d'anneau avec identité $R = (S, +, \times)$:

- G est associatif, commutatif, possède un élément neutre et des inverses,
- R est associatif $(+, \times)$, commutatif $(+)$, possède un élément neutre $(+, \times)$ et des inverses $(+)$.

L'espace vectoriel possède également un produit entre les él'ements de S et ceux de V tel que:

- $c(\mathbf{u} \oplus \mathbf{v}) = (c\mathbf{u}) \oplus (c\mathbf{v})$,
- $(b + c)\mathbf{v} = (b\mathbf{v}) \oplus (c\mathbf{v})$,
- $(b \times c)\mathbf{v} = b(c\mathbf{v})$,
- $1\mathbf{v} = \mathbf{v}$ où 1 est l'élément neutre multiplicatif de R.
- $0\mathbf{v} = \mathbf{0}$ où 0 est l'élément neutre additif de R et $\mathbf{0}$ l'élément neutre de G.

Exemple: On défini un espace vectoriel sur un corps fini $\mathcal{F}_q$ en prennant $V = \{(e_1, e_2, ..., e_n) : e_i \in \mathcal{F}_q\}$ et $S = \mathcal{F}_q$.

3.1 Combinaisons linéaires

Une *combinaison linéaire* est une somme de la forme $c_1\mathbf{v}_1 \oplus c_2\mathbf{v}_2 \oplus ... \oplus c_n\mathbf{v}_n$. Un ensemble de vecteurs $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\}$ sont *linéairement dépendent* s'il existent des scalaires $c_1, c_2, ..., c_n$ non tous nuls tels que $c_1\mathbf{v}_1 \oplus c_2\mathbf{v}_2 \oplus ... \oplus c_n\mathbf{v}_n = \mathbf{0}$. Sinon ils sont *linéairement indépendent.*

3.2 Produit Scalaire

Le produit scalaire $(\mathbf{u} \cdot \mathbf{v})$ de deux vecteurs $\mathbf{u} = (u_1, u_2, ..., u_n)$ et $\mathbf{v} = (v_1, v_2, ..., v_n)$ est défini par

$$\mathbf{u} \cdot \mathbf{v} = u_1 \times v_1 + u_2 \times v_2 + ... + u_n \times v_n.$$

Deux vecteurs tels que $(\mathbf{u} \cdot \mathbf{v}) = 0$ sont *orthogonaux.* Contrairement aux nombres réels où le seul vecteur orthogonal à lui même est le vecteur $\mathbf{0}$, sur un corps fini il peut exister plusieurs vecteurs orthogonaux à eux même.

Exemple: Les vecteurs $(1, 0, 0, 1)$ et $(1, 1, 1, 1)$ sont auto-orthogonaux sur $\mathcal{F}_2$ car $(1, 0, 0, 1) \cdot (1, 0, 0, 1) = (1, 1, 1, 1) \cdot (1, 1, 1, 1) = 0$.

4 Algèbre Linéaire

On défini des matrices sur $\mathcal{F}_q$ comme sur les nombres réels. Une *matrice* $n \times m$ (n par m) sur $\mathcal{F}_q$ est de la forme suivante où chaque $a_{ij} \in \mathcal{F}_q$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$

Note: Un vecteur $\mathbf{v}$ peut être considéré comme une matrice contenant une seule ligne.

4.1 opérations

La somme de deux matrices $\mathbf{A}, \mathbf{B}$ $n \times m$ est défini par

$$\mathbf{A}+\mathbf{B} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1m}+a_{1m} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2m}+b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}+b_{n1} & a_{n2}+b_{n2} & \dots & a_{nm}+b_{nm} \end{bmatrix}.$$

Le produit d'un scalaire et d'une matrice $\mathbf{A}$ $n \times m$ est défini par

$$c\mathbf{A} = \begin{bmatrix} c \times a_{11} & c \times a_{12} & \dots & c \times a_{1m} \\ c \times a_{21} & c \times a_{22} & \dots & c \times a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c \times a_{n1} & c \times a_{n2} & \dots & c \times a_{nm} \end{bmatrix}.$$

Le produit de deux matrices $\mathbf{A}, \mathbf{B}$ respectivement $n \times k$ et $k \times m$ est défini par

$$\mathbf{AB} = \begin{bmatrix} \mathbf{A}_{1*} \cdot \mathbf{B}_{*1} & \mathbf{A}_{1*} \cdot \mathbf{B}_{*2} & \dots & \mathbf{A}_{1*} \cdot \mathbf{B}_{*m} \\ \mathbf{A}_{2*} \cdot \mathbf{B}_{*1} & \mathbf{A}_{2*} \cdot \mathbf{B}_{*2} & \dots & \mathbf{A}_{2*} \cdot \mathbf{B}_{*m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{n*} \cdot \mathbf{B}_{*1} & \mathbf{A}_{n*} \cdot \mathbf{B}_{*2} & \dots & \mathbf{A}_{n*} \cdot \mathbf{B}_{*m} \end{bmatrix}$$

où $\mathbf{A}_{i*}$ est la $i^{\text{ième}}$ ligne de $\mathbf{A}$ et $\mathbf{B}_{*j}$ est la $j^{\text{ième}}$ colonne de $\mathbf{B}$.

4.2 temps

L'addition de deux matrices $n \times m$ prend $O(nm)$ additions du corps fini. Le produit d'un scalaire avec une matrice $n \times m$ prend $O(nm)$ multiplications du corps fini. Le produit de deux matrices $n \times k$, $k \times m$ prend $O(nmk)$ multiplications du corps fini si implémenté de façon "naturelle". Si implémenté de façon plus subtile on peut réduire ce temps de calcul pour des grandes matrices. Dans le cas où $n = m = k$ on peut multiplier deux matrices par $O(n^{2.376})$ multiplications du corps fini au lieu de $O(n^3)$. Consulter "Fundamentals of Algorithmics" par Brassard et Bratley pour plus de détails.

4.3 déterminant

Le *déterminant* d'une matrice $\mathbf{A}$ carrée $n \times n$ est défini par

$$det(\mathbf{A}) = \sum_{(i_1,i_2,\dots,i_n)\in\pi(1,2,\dots,n)} -1^{\sigma(i_1,i_2,\dots,i_n)} a_{1i_1} \times a_{2i_2} \times \dots \times a_{ni_n}$$

où $\sigma(i_1, i_2, ..., i_n)$ est la parité de la permutation $(i_1, i_2, ..., i_n)$ et $\pi(1, 2, ..., n)$ l'ensemble des permutations de $1, 2, ..., n$.

On utilise le déterminant pour décider si une matrice est constituée de vecteurs linéairement dépendents ($\det(\mathbf{A}) = 0$) ou non ($\det(\mathbf{A}) \neq 0$).

Temps: Si on utilise bêtement cette définition pour calculer le déterminant il en découle un algorithme qui prend un temps $n!$, ce qui est très mauvais. C'est aussi mauvais si l'on utilise le formule récursive bien connue. Si l'on utilise l'algorithme de Gauss-Jordan qui consiste à d'abord réduire la matrice pour rendre le calcul du déterminant trivial, il en résulte un algorithme en temps $O(n^3)$. On peut faire mieux en utilisant des lagorithmes de multiplication rapide et descendre l'exposant à au plus $O(n^{\log_2 7}) \approx O(n^{2.81})$.

4.4 matrices échelonées réduites

L'espace linéaire *engendré* par une matrice **A** est l'ensemble des combinaisons linéaires de ses lignes. On peut faire trois types d'opérations à la matrice **A** qui ne modifie pas l'espace qu'elle engendre:

- permuter deux lignes,
- multiplier une ligne par un scalaire non nul,
- remplacer une ligne par elle-même à laquelle on ajoute un scalaire fois une autre ligne.

Il est souvent souhaitable de changer de matrice, tout en gardant le même espace engendré, et de l'avoir selon une forme dite *échelonée réduite*:

- la première valeur non nulle de chaque ligne est 1,
- les autres lignes contiennent 0 dans cette colonne,
- la première valeur non nulle de chaque ligne est à droite de celles des lignes plus hautes dans la matrice.

5 Codes

Un (n, M, d) code C sur $\mathcal{F}_q$ est un ensemble de M mots de $\mathcal{F}_q^n$ où d est la distance minimale entre deux mots de C au sens de Hamming, $d = \min\{\Delta(c_1, c_2) : c_1, c_2 \in C\}$. La distance de Hamming entre deux mots c_1, c_2, dénotée $\Delta(c_1, c_2)$, est le nombre de positions où c_1 et c_2 diffèrent.

5.1 taux de transmission d'un code

Le *taux de transmission* d'un (n, M, d) code est le ratio entre la longeur nécessaire pour exprimer les M mots de code de façon minimale et n:

$$R = \frac{\log_q M}{n}.$$

5.2 distance minimale

Lors de la transmission d'un mot de code $c \in C$ sur un canal imparfait, il est TOUJOURS possible de corriger $\lfloor (d-1)/2 \rfloor$ erreurs, en choisissant le mot de code c^* le plus près du mot reçu w. Si le nombre d'erreur est inférieur ou égal à $(d-1)/2$ il ne peut y avoir d'ambigüité sur le mot de code original car il ne peut exister deux mots de codes $c_1 \neq c_2$ tels que $\Delta(w, c_1) \leq (d-1)/2$ et $\Delta(w, c_2) \leq (d-1)/2$ étant donné que ces deux mots satisferait alors $\Delta(c_1, c_2) \leq d-1$. On appèlle la *sphère* associée à un mot de code c l'ensemble des mots à distance inférieur ou égal à $(d-1)/2$ de c:

$$\Phi_c = \{w \in \mathcal{F}_q^n : \Delta(w, c) \leq (d-1)/2\}.$$

5.3 codes parfaits

Un code *parfait* est un code où tous les mots de $\mathcal{F}_q^n$ sont dans la sphère d'un mot de code.

Theorem 7. *Un code (n, M, d) est parfait si et seulement si d est impair et*

$$M \sum_{i=0}^{(d-1)/2} \binom{n}{i} (q-1)^i = q^n.$$

Exemple: Le code $C = \{00000...0, 11111...1\}$ où les mots sont de longueur impaire est parfait.

5.4 codes équivalents

Deux codes $C, C' \subseteq \mathcal{F}_q^n$ sont *équivalents* s'il existe une permutation π des positions $1, 2, ..., n$ telle que

$$(c_1, c_2, ..., c_n) \in C \iff (c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(n)}) \in C'.$$

Theorem 8. *Si C est un code (n, M, d), C' est un code (n, M', d') et C, C' sont équivalent alors $M = M'$ et $d = d'$*

6 Codes Linéaires

Un $[n, k, d]$ code linéaire C sur $\mathcal{F}_q$ est un (n, q^k, d) code qui satisfait la propriété que si $\mathbf{c}_1, \mathbf{c}_2 \in C$ alors $\alpha\mathbf{c}_1 + \beta\mathbf{c}_2 \in C$ pour $\alpha, \beta \in \mathcal{F}_q$. Ou encore, C est un sous-espace vectoriel de dimension k de $\mathcal{F}_q^n$. Les codes linéaires ont de nombreux avantages au niveau de leur représentation concise (il suffit de k mots de code pour specécifier une base de C) et leur codage/décodage efficace.

6.1 distance minimale

La distance minimale d'un code linéaire est donnée par $d = \min\{w(\mathbf{c}) : \mathbf{c} \in C - \{0^n\}\}$ où $w(\mathbf{c})$ est le poids de $\mathbf{c}$, c'est-à-dire le nombre de positions non-nulles de $\mathbf{c}$.

6.2 matrice génératrice

Un $[n,k,d]$ code linéaire C sur $\mathcal{F}_q$ est spécifié par une *matrice génératrice* $\mathbf{G}$ de dimensions $k \times n$ de rang k telle que

$$C = \text{ espace engendré par les lignes de } \mathbf{G}.$$

Exemple: la matrice suivante sur $\mathcal{F}_3$ $\begin{bmatrix} 2\,0\,1 \\ 0\,0\,1 \end{bmatrix}$ engendre le code

$$C = \{(0\ 0\ 0), (0\ 0\ 1), (0\ 0\ 2), (2\ 0\ 1), (1\ 0\ 2), (2\ 0\ 2), (1\ 0\ 0), (2\ 0\ 0), (1\ 0\ 1)\}.$$

6.3 codage et décodage

La matrice génératrice $\mathbf{G}$ nous donne une application de $\mathcal{F}_q^k \to \mathcal{F}_q^n$ qui défini le codage d'un mot $\mathbf{w}$ de $\mathcal{F}_q^k$ par un mot de code $\mathbf{c}$ de $\mathcal{F}_q^n$ par

$$\mathbf{c} = \mathbf{w}\mathbf{G}.$$

Le décodage d'un mot de code $\mathbf{c}$ est le mot $\mathbf{w}$ qui satisfait l'équation précédente.

6.4 forme systématique

Une matrice $\mathbf{G}$ est en *forme systématique* si $\mathbf{G} = [\mathbf{I}_k|\mathbf{A}]$ où $\mathbf{I}_k$ est la matrice identité $k \times k$ et $\mathbf{A}$ une matrice $k \times (n-k)$ quelconque. Un code C possède une forme systématique s'il existe une matrice $\mathbf{G}$ en forme systématique qui l'engendre.

Theorem 9. *Tout $[n,k,d]$ code C est équivalent à un $[n,k,d]$ code C' qui possède une forme systématique.*

Avantages: Quand on a un code C spécifié par un matrice $\mathbf{G}$ en forme systématique le codage et le décodage sont simplement sécifiés par l'équation

$$\mathbf{w}\mathbf{G} = [\mathbf{w}|\mathbf{w}\mathbf{A}].$$

Pour coder un mot $\mathbf{w}$ il suffit donc de lui adjoindre $\mathbf{w}\mathbf{A}$ et pour décoder un mot de code $\mathbf{c} = [\mathbf{w}|\mathbf{x}]$ il suffit d'extraire $\mathbf{w}$.

6.5 matrice de parité

Un $[n,k,d]$ code linéaire C, peut également spécifié par une *matrice de parité* $\mathbf{H}$ de dimensions $(n-k) \times n$ de rang $n-k$ telle que

$$C = \{\mathbf{c} \in \mathcal{F}_q^n : \mathbf{c}\mathbf{H}^\top = \mathbf{0}_{n-k}\}$$

où $\mathbf{0}_{n-k}$ est le vecteur nul de $\mathcal{F}_q^{n-k}$. Une relation simple relie la matrice génératrice $\mathbf{G}$ et la matrice de parité $\mathbf{H}$:

$$\mathbf{G}\mathbf{H}^\top = \mathbf{0}_{k\times(n-k)} \qquad \mathbf{H}\mathbf{G}^\top = \mathbf{0}_{(n-k)\times k}.$$

Notons que pour un $\mathbf{G}$ donné, $\mathbf{H}$ n'est pas unique et vice vera. Néanmoins i $\mathbf{G}$ est en forme systématique $[\mathbf{I}_k|\mathbf{A}]$, il est alors trivial de trouver un $\mathbf{H}$ correspondant en prenant

$$\mathbf{H} = [-\mathbf{A}^\top|\mathbf{I}_{n-k}].$$

6.6 syndromes et cosets

On utilise la matrice $\mathbf{H}$ pour définir une fonction de $\mathcal{F}_q^n \to \mathcal{F}_q^{n-k}$ que l'on nomme *syndrome* d'un mot $\mathbf{w} \in \mathcal{F}_q^n$:

$$Syn(\mathbf{w}) = \mathbf{w}\mathbf{H}^\top.$$

On appele *coset* l'ensemble des mots de même syndrome

$$Coset(\mathbf{w}) = \{\mathbf{x} \in \mathcal{F}_q^n : Syn(\mathbf{w}) = Syn(\mathbf{x})\}.$$

On note que $C = Coset(\mathbf{0})$ et que $Coset(\mathbf{w}) = \{\mathbf{x} \in \mathcal{F}_q^n : \mathbf{x} = \mathbf{w} + \mathbf{c}, \mathbf{c} \in C\}$.

6.7 détection d'erreurs

Dans un code linéaire il est très simple de détecter des erreurs puisqu'un mot reçu w est erroné si et seulement si $\mathbf{w}\mathbf{H}^\top \neq \mathbf{0}_{n-k}$. Si une erreur est assez importante pour provoquer qu'un mot de code soit projeté sur un autre mot de code, il n'y a aucune façon de le détecter.

6.8 correction d'erreurs

La correction d'erreur dépend généralement du modèle d'erreurs qui régit notre canal de communication imparfait, mais dans la plupart des cas les vecteurs d'erreur de poids inférieurs sont plus probable que ceux de poids supérieurs. Ceci induit le décodage à distance minimale, comme méthode de correction d'erreurs. Il s'agit de trouver le mot de code ayant la plus petite distance au mot reçu

$$corrige(\mathbf{w}) = \mathbf{c} \text{ t.q. } \Delta\left(\mathbf{w}, \mathbf{c}\right) \leq \Delta\left(\mathbf{w}, \mathbf{c'}\right), \mathbf{c}' \in C.$$

Dans un code parfait, ce mot de code est défini de façon unique pour tous les mots possible.

En général, le probléme de la correction d'erreur est équivalent à celui de trouver un mot de poids minimum d'un certain syndrome.

Theorem 10. *Étant donné un mot quelconque* $\mathbf{w}$*, soit* $\mathbf{w}^*$ *un mot de poids minimal dans* $Coset(\mathbf{w})$*. Le mot* $\mathbf{w} - \mathbf{w}^*$ *est un mot de code à distance minimale de* $\mathbf{w}$*.*

En général, on ne connait pas de façon efficace de trouver un mot de poids minimal d'un certain syndrome. Si l'on énumère tous les mots d'un coset pour trouver celui de poids minimal, on utilisera un temps $O(q^k)$. Si le poids minimal est relativement grand, une recherche exhaustive à partir des mots de poids $0, 1, 2, \ldots$ prendre également un temps exponentiel. On verra bientôt que ce problème est NP-complet.

6.9 code dual

À chaque $[n, k, d]$ code linéaire C correspond un $[n, n-k, \hat{d}]$ code *dual* $\hat{C}$ engendré par la matrice $\mathbf{H}$. La matrice $\mathbf{G}$ est sa matrice de parité. On ne connait que des relation triviales entre d et $\hat{d}$. On verra plus tard des applications où on considèrera ces deux codes simultanément.

7 Codes de Hamming

Les codes de Hamming forment une famille infinie $\{\mathcal{H}_m\}_{m>0}$ de codes linéaires binaires ayant des paramètres $[2^m-1, 2^m-m-1, 3]$.

7.1 construction

La matrice de parité du code $\mathcal{H}_m$ est

$$\mathbf{H}_m = [\boldsymbol{\beta}_m^\top(1)|\boldsymbol{\beta}_m^\top(2)|...|\boldsymbol{\beta}_m^\top(2^m-1)]$$

où $\boldsymbol{\beta}_m(i)$ est le vecteur binaire de longueur m représentant le nombre entier i. Exemple: $\boldsymbol{\beta}_4(5) = (0\ 1\ 0\ 1)$.

$$\mathcal{H}_3 = \begin{bmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{bmatrix}$$

7.2 distance minimale

La distance minimale de tels codes est 3. Supposons au contraire qu'elle soit 1 ou 2. Considérons d'abord un mot $\mathbf{w}$ de poids 1. Le produit $\mathbf{w}\mathbf{H}^\top = \boldsymbol{\beta}_m(i)$ où i est la position du bit non-nul de $\mathbf{w}$. Comme aucun $\boldsymbol{\beta}_m(i)$ n'est nul, il en découle qu'aucun $\mathbf{w}$ de poids 1 ne peut être un mot de code. Considérons similairement un mot $\mathbf{w}$ de poids 2. Le produit $\mathbf{w}\mathbf{H}^\top = \boldsymbol{\beta}_m(i) \oplus \boldsymbol{\beta}_m(j)$ où i, j sont les positions des bits non-nuls de $\mathbf{w}$. Comme aucun $\boldsymbol{\beta}_m(i) = \boldsymbol{\beta}_m(j)$, il en découle que $\boldsymbol{\beta}_m(i) \oplus \boldsymbol{\beta}_m(j)$ ne peut être nul, et donc qu'aucun $\mathbf{w}$ de poids 2 ne peut être un mot de code. Par contre, le mot $(1\ 1\ 1\ \underbrace{0\ 0\ 0\ ...\ 0}_{2^m-4})$ de poids 3 est toujours un mot de code car son syndrome est $\boldsymbol{\beta}_m(1) \oplus \boldsymbol{\beta}_m(2) \oplus \boldsymbol{\beta}_m(3) = \mathbf{0}$.

On remarque que $\mathcal{H}_m$ est un code parfait puisque

$$M \sum_{i=0}^{(d-1)/2} \binom{n}{i} (q-1)^i = 2^{2^m-m-1} \sum_{i=0}^{1} \binom{2^m-1}{i} (1)^i = 2^{2^m-m-1}(1+2^m-1) = 2^{2^m-1} = q^n.$$

7.3 correction d'erreurs

Un code de Hamming permet de corriger exactement une erreur. C'est de toute façon le mieux que l'on puisse s'attendre de ce code puisqu'il est parfait et que $(d-1)/2 = 1$. Soit $\mathbf{c}$ un mot de code transmis et $\mathbf{w} = \mathbf{c} + \mathbf{e}$ le mot reçu, pour un certain mot d'erreur $\mathbf{e}$ de poids 1. Comme on le remarquait plus haut, le syndrome d'un mot de poids 1 est $\boldsymbol{\beta}_m(i)$ où i est la position du bit non-nul de $\mathbf{e}$. Puisque $\mathbf{w} = \mathbf{c} + \mathbf{e}$, $\mathbf{w}$ et $\mathbf{e}$ on le même syndrome car selui de $\mathbf{c}$ est nul. Il en découle que $Syn(\mathbf{w})$ sera la représentation binaire de la position où l'erreur s'est produite. Pour corriger $\mathbf{w}$ il suffira donc d'inversion le bit à la position indiquée par le syndrome.

8 Théorie de l'information

La théorie de l'information formalise les notions d'*incertitude* et d'*information* de façon mathématique. On consère pour cela des variables aléatoires, que l'on nommera par des lettres majuscules ex: $X, Y, Z, f(X), \ldots$ Rappelons qu'une variable aléatoire est un objet stochastique pouvant prendre des valeurs $x_1, x_2, \ldots, x_n$ avec probabilités $p_1, p_2, \ldots, p_n$ telles que $\sum_{i=1}^n p_i = 1$. On ne considère ici que les variables aléatoire prenanant qu'un nombre fini de valeurs. On dénotera $P[X = x]$ ou $P_X(x)$ la probabilité que la variable aléatoire X prenne la valeur x:

$$P_X(x) = P[X = x].$$

Rappelons que si l'on utilise deux ou plusieurs variables aléatoire (X, Y) distribuées selon $P_{X,Y}$, la *probabilité marginale* par rapport à X est donnée par

$$P_X(x) = \sum_y P_{X,Y}(x, y)$$

et que la *probabilité conditionnelle* de X par rapport à Y est donnée par

$$P[X = x | Y = y] = P_{X|Y=y}(x) = P_{X,Y}(x, y)/P_Y(y).$$

Afin d'alléger la notation on remplacera les expressions du type $1 - \epsilon$ par $\bar{\epsilon}$ pour une valeur rélle ϵ et l'on définit la fonction h par:

$$h(\epsilon) = -\epsilon \lg \epsilon - \bar{\epsilon} \lg \bar{\epsilon}, \quad \text{avec } \bar{\epsilon} = 1 - \epsilon.$$

8.1 Modèles d'erreurs

On considere principalement deux modèles d'erreurs: le canal binaire symétrique et le canal d'effacement. Le premier (BS_ϵ) est un canal où une valeur de sortie est toujours disponible, mais elle sera erronnée avec probabilité ϵ.

$$\begin{aligned} P[BS_\epsilon(x) = x] &= \bar{\epsilon} \\ P[BS_\epsilon(x) \neq x] &= \epsilon \end{aligned}$$

Dans le cas binaire, puisqu'il n'y a que deux possibilités pour la valeur de x on aura $P[BS_\epsilon(x) = \bar{x}] = P[BS_\epsilon(x) \neq x]$. Dans un cas non-binaire, sur un corps $\mathcal{F}_q$ pour $q > 2$ par exemple, on suppose une distribution uniforme des résultats erronnés, c'est à dire que $P[BS_\epsilon(x) \neq x]$ est partagée également entre les $y \neq x$:

$$\begin{aligned} P[BS_\epsilon(x) = x] &= \bar{\epsilon} \\ P[BS_\epsilon(x) = y] &= \tfrac{\epsilon}{q-1} \text{ pour } y \neq x \end{aligned}$$

Le second canal ne donne jamais de résultat faux, mais parfois aucun résultat n'est obtenu en sortie:

$$\begin{aligned} P[EF_\epsilon(x) = x] &= \bar{\epsilon} \\ P[EF_\epsilon(x) = \varepsilon] &= \epsilon \end{aligned}$$

Généralisation Si l'on généralise ces deux canaux pour transmettre des mots de n symboles de façon indépendante par l'intermédiaire des canaux définis précédemment on obtient le canal binaire symétrique d'ordre n (BS_ϵ^n) et le canal d'effacement d'ordre n (EF_ϵ^n):

$$P[BS_\epsilon^n(w) = w'] = \bar{\epsilon}^{n - \Delta(w,w')} \left(\frac{\epsilon}{q-1}\right)^{\Delta(w,w')}$$

$$P[EF_\epsilon^n(w) = w'] = \bar{\epsilon}^{n - \varepsilon(w')} \epsilon^{\varepsilon(w')}$$

où $\varepsilon(w') = \#\{i : w_i' = \varepsilon\}$.

8.2 Entropie

$$H(p_1, p_2, ..., p_n) = -\sum_{i=1}^{n} p_i \lg p_i$$

propriétés

1. $H(p_1, p_2, ..., p_n)$ est maximum pour $p_1 = p_2 = ... = p_n = 1/n$,
2. $H(p_1, p_2, ..., p_n) = H(p_{\pi(1)}, p_{\pi(2)}, ..., p_{\pi(n)})$ pour toute permutation π,
3. $H(p_1, p_2, ..., p_n) \geq 0$ et $H(p_1, p_2, ..., p_n) = 0$ exactement si un $p_i = 1$,
4. $H(p_1, p_2, ..., p_n, 0) = H(p_1, p_2, ..., p_n)$,
5. $H(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}) \leq H(\frac{1}{n+1}, \frac{1}{n+1}, ..., \frac{1}{n+1}, \frac{1}{n+1})$,
6. H est une fonction continue,
7. $H(\frac{1}{mn}, \frac{1}{mn}, ..., \frac{1}{mn}) = H(\frac{1}{m}, \frac{1}{m}, ..., \frac{1}{m}) + H(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n})$,
8. $H(p_1, p_2, ..., p_m, q_1, q_2, ...q_n) = H(p, q) + pH(\frac{p_1}{p}, \frac{p_2}{p}, ..., \frac{p_m}{p}) + qH(\frac{q_1}{q}, \frac{q_2}{q}, ..., \frac{q_n}{q})$ où $p = \sum p_i$ et $q = \sum q_i$.

Theorem 11. *Soit une fonction $H'(p_1, p_2, ..., p_n)$ satisfaisant les propriétés 1. à 8. ci-haut. Il existe toujours une constante positive λ telle que*

$$H'(p_1, p_2, ..., p_n) = \lambda H(p_1, p_2, ..., p_n).$$

Theorem 12.

$$H(p_1, p_2, ..., p_n) \leq \lg n.$$

Theorem 13. *Pour toutes distribution de probabilité $q_1, q_2, ..., q_n$ on a*

$$-\sum_{i=1}^{n} p_i \lg q_i \leq -\sum_{i=1}^{n} p_i \lg p_i.$$

On définit enfin l'*entropie* d'une variable aléatoire X pouvant prendre les valeurs $x_1, x_2, ..., x_n$ par

$$\mathbf{H}(X) = H(P_X(x_1), P_X(x_2), ..., P_X(x_n)) = -\sum_x P_X(x) \lg P_X(x)$$

Theorem 14.

$$\mathbf{H}(X, Y) \leq \mathbf{H}(X) + \mathbf{H}(Y)$$

avec égalité exactement si X et Y sont indépendantes.

Theorem 15.

$$\mathbf{H}(X_1, X_2, ..., X_n) \leq \mathbf{H}(X_1) + \mathbf{H}(X_2) + ...\mathbf{H}(X_n)$$

avec égalité exactement si $X_1, X_2, ..., X_n$ sont mutuellement indépendantes.

8.3 Entropie conditionelle

$$\mathbf{H}(X|Y) = \sum_y P_Y(y)\mathbf{H}(X|Y = y) = -\sum_y \sum_x P_Y(y) P_{X|Y=y}(x) \lg P_{X|Y=y}(x)$$

Theorem 16.

$$\mathbf{H}(X|Y) = 0$$

exactement si $X = f(Y)$ pour une certaine fonction f et

$$\mathbf{H}(X|Y) = \mathbf{H}(X)$$

lorsque X et Y sont indépendantes.

Theorem 17.

$$\mathbf{H}(X, Y) = \mathbf{H}(Y) + \mathbf{H}(X|Y)$$

et donc

$$\mathbf{H}(X|Y) \leq \mathbf{H}(X)$$

avec égalité lorsque X et Y sont indépendantes.

8.4 Information mutuelle

$$\mathbf{I}(X;Y) = \mathbf{H}(X) - \mathbf{H}(X|Y)$$

Theorem 18.

$$\mathbf{I}(X;Y) = \mathbf{I}(Y;X).$$

Exemple:

option vs vote	OUI	NON	ind.
séparation	30%	60%	10%
offre de partenariat	39%	46%	15%
certitude de partenariat	54%	35%	11%

Fig. 3. Table de $P[VOTE = vote|OPTION = option]$

$$P[OPTION = \text{“séparation”}] = 1/4$$

$$P[OPTION = \text{“offre de partenariat”}] = 1/2$$

$$P[OPTION = \text{“certitude de partenariat”}] = 1/4$$

$$P[VOTE = \text{“OUI”}] = 40.5\%$$

$$P[VOTE = \text{“NON”}] = 46.75\%$$

$$P[VOTE = \text{“ind.”}] = 12.75\%$$

$$\mathbf{H}(VOTE) = -.405 \lg .405 - .4675 \lg .4675 - .1275 \lg .1275 = 1.4198$$

$$\mathbf{H}(OPTION) = -.25 \lg .25 - .5 \lg .5 - .25 \lg .25 = 1.5$$

$$\mathbf{H}(VOTE|OPTION = \text{“séparation”}) = -.3 \lg .3 - .6 \lg .6 - .1 \lg .1 = 1.2955$$

$$\mathbf{H}(VOTE|OPTION = \text{“offre de partenariat”}) = -.39 \lg .39 - .46 \lg .46 - .15 \lg .15 = 1.4557$$

$$\mathbf{H}(VOTE|OPTION = \text{“certitude de partenariat”}) = -.54 \lg .54 - .35 \lg .35 - .11 \lg .11 = 1.3604$$

$$\mathbf{H}(VOTE|OPTION) = \frac{1.2955}{4} + \frac{1.4557}{2} + \frac{1.3604}{4} = 1.3918$$

$$\mathbf{I}(VOTE;OPTION) = \mathbf{H}(VOTE) - \mathbf{H}(VOTE|OPTION) = 1.4198 - 1.3918 = 0.0280$$

8.5 Capacité d'un canal

Un canal de communication σ est une fonction stochastique qui associe à chaque élément x de son domaine une variable aléatoire $\sigma(x)$. Il est naturel de se demander combien d'information un canal σ transmet au sujet d'une variable aléatoire X qui lui est fournie comme donnée. Cette valeur est $\mathbf{I}(X;\sigma(X))$. Il est clair que plus cette valeur est grande, plus le canal communique beaucoup sur la source. On définit la *capacité* d'un canal σ par la quantité suprémum d'information qu'il peut transmettre pour toutes les variables aléatoires que l'on pourrait lui fournir en entrée:

$$\mathbf{C}(\sigma) = \sup_X \mathbf{I}(X;\sigma(X)).$$

Theorem 19.

$$\mathbf{C}(BS_\epsilon) = 1 - h(\epsilon).$$

Theorem 20.

$$\mathbf{C}(BS_\epsilon^n) = n\mathbf{C}(BS_\epsilon) = n - nh(\epsilon).$$

8.6 Objectifs de la théorie des codes

Pour tout code C on définit la probabilité d'erreur du code par

$$e(C) = \frac{1}{M}\sum_{i=1}^{M} P[\text{erreur}|c_i \text{ transmis}].$$

$$\hat{e}(C) = \max_{i=1}^{M} P[\text{erreur}|c_i \text{ transmis}].$$

La théorie des codes a pour but de construire des familles de codes $\{C_n\}_{n>0}$ de paramètres (n, M_n, d_n) permettant de transmettre des informations au meilleur coût possible en obtenant le moins d'erreur possible à travers un BS_ϵ. On voudrait optimiser deux paramètres à la fois $R_n = \lg M_n/n$ et $\hat{e}(C_n)$. En fait, on cherche le meilleur R pour lequel il existe une constante positive $\alpha < 1$ telle que

$$\lim_{n\to\infty} R_n = R \quad \text{et} \quad \hat{e}(C_n) < \alpha^n.$$

8.7 Théorèmes de Shannon

Theorem 21. *Soient ϵ et δ, tels que $0 < \delta, \epsilon < 1$ et R tel que $0 < R < \mathbf{C}(BS_\epsilon)$. Pour tout ensemble $\{M_n\}_{n>0}$, tels que $0 < M_n \leq 2^{Rn}$, il existe un entier n_0 et une famille de codes $\{C_n\}_{n>n_0}$ de paramètres (n, M_n, d_n) tels que $\hat{e}(C_n) < \delta$ pour tout $n > n_0$.*

Theorem 22. *Soit ϵ, tel que $0 < \epsilon < 1$ et soit R tel que $0 < R < \mathbf{C}(BS_\epsilon)$. Il existe un entier n_0, une constante positive $\alpha < 1$ et une famille infinie de codes $\{C_n\}_{n>n_0}$ de paramètres (n, M_n, d_n) tels que*

$$M_n = \lceil 2^{Rn} \rceil \quad \textit{et} \quad \hat{e}(C_n) < \alpha^n$$

pour tout $n > n_0$.

8.8 Outils utiles

Loi des grands nombres de Bernstein

Theorem 23. *Soient $X_1, X_2, ..., X_n$ des variables aléatoires de Bernoulli indépendantes telles que $P[X_i = 1] = \epsilon$ pour $1 \leq i \leq n$. Pour tout δ, $0 < \delta \leq \epsilon(1-\epsilon)$ on a*

$$P\left[\left|\sum_{i=1}^{n} \frac{X_i}{n} - \epsilon\right| \geq \delta\right] \leq 2e^{-n\delta^2}.$$

Approximation d'une binomiale

Theorem 24.
$$\frac{2^{h(\lambda)n}}{\sqrt{8n\lambda(1-\lambda)}} \leq \binom{n}{\lambda n} \leq \frac{2^{h(\lambda)n}}{\sqrt{2\pi n\lambda(1-\lambda)}}.$$

Approximation de la somme des binomiales

Theorem 25. $\sum_{i=0}^{\lambda n} \binom{n}{i} \leq 2^{h(\lambda)n}$.

Preuve:

$$\begin{aligned} 1 = [\lambda + (1-\lambda)]^n &\geq \sum_{i=0}^{\lambda n} \binom{n}{i} \lambda^i (1-\lambda)^{n-i} \\ &\geq (1-\lambda)^n \sum_{i=0}^{\lambda n} \binom{n}{i} \left(\frac{\lambda}{1-\lambda}\right)^{\lambda n} \\ &= \lambda^{\lambda n}(1-\lambda)^{n(1-\lambda)} \sum_{i=0}^{\lambda n} \binom{n}{i} \end{aligned}$$

Theorem 26 Elias. *Soit ϵ, tel que $0 < \epsilon < 1$ et soit R tel que $0 < R < \mathbf{C}(BS_\epsilon)$. Il existe un entier n_0, une constante positive $\alpha < 1$ et une famille infinie de codes* **linéaires** $\{C_n\}_{n>n_0}$ *de paramètres $[n, k_n, d_n]$ tels que*
$$\lim_{n\to\infty} \frac{k_n}{n} = R \quad \textit{et} \quad \hat{e}(C_n) < \alpha^n$$
pour tout $n > n_0$.

9 Limites sur les performances des codes

9.1 Borne supérieure de Singleton

Theorem 27 Borne de Singleton. *Pour tout code (n, M, d) on a $q^{n-d+1} \geq M$ et en particulier pour tout code linéaire $[n, k, d]$*

$$n - k \geq d - 1.$$

9.2 Borne inférieure de Varshamov-Gilbert

Theorem 28. *Soit* $\mathbf{H}$ *la matrice de parité d'un code linéaire de longueur n. La dimension de ce code est $n - r$ où r est le nombre de colonnes linéairement indépendantes de* $\mathbf{H}$.

Theorem 29. *Soit* $\mathbf{H}$ *la matrice de parité d'un code linéaire de longueur n. La distance minimale d de ce code est telle que tout ensemble de $d - 1$ colonnes de* $\mathbf{H}$ *sont linéairement indépendantes mais il existe d colonnes de* $\mathbf{H}$ *linéairement dépendantes.*

Soit $h_q(x) = x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x)$.

Theorem 30 Borne de Varshamov-Gilbert. *Soit D_q le plus grand entier tel que*

$$\binom{n-1}{0} + \binom{n-1}{1}(q-1) + \binom{n-1}{2}(q-1)^2 + ... + \binom{n-1}{D_q-2}(q-1)^{D_q-2} < q^{n-k}$$

alors il existe sur $\mathcal{F}_q$ un code linéaire $[n, k, d]$ pour $d \geq D_q$ et asymptotiquement, il existe un code linéaire $[n, \rho n, d]$ pour $d \geq \epsilon n$ si

$$\rho < 1 - h_q(\epsilon).$$

Theorem 31 Borne de Varshamov-Gilbert. *Soit ϵ et δ des constantes réelles. Soit $\rho = 1 - h_q(\epsilon)$. Soit C le $[n, \rho n, d]$ code sur $\mathcal{F}_q$ engendré par une matrice aléatoire de rang ρn et de dimensions $\rho n \times n$. Il existe un $\alpha < 1$ tel que*

$$\mathrm{P}\left(d > (\epsilon - \delta)n\right) \geq 1 - \alpha^n$$

et

$$\mathrm{P}\left(d > (\epsilon + \delta)n\right) \leq \alpha^n.$$

9.3 Borne supérieure de Hamming

Theorem 32 Borne de Hamming. *Il n'existe pas sur $\mathcal{F}_q$ de code $(n, M, 2t)$ ou $(n, M, 2t+1)$ dès que*

$$\binom{n-1}{0} + \binom{n-1}{1}(q-1) + \binom{n-1}{2}(q-1)^2 + ... + \binom{n}{t}(q-1)^t > q^n/M$$

et asymptotiquement, il n'existe pas de code $(n, q^{\rho n}, d)$ pour $d \geq \epsilon n$ si

$$\rho > 1 - h_q(\epsilon/2).$$

9.4 Borne supérieure de Plotkin

Theorem 33 Borne de Plotkin. *Pour tout code (n, M, d) pour lequel $d > \theta n = \frac{q-1}{q} n$ on a*

$$M \leq \frac{d}{d - \theta n}.$$

9.5 Borne supérieure de McEliece-Rodemich-Rumsey-Welch

Theorem 34 Borne de McEliece-Rodemich-Rumsey-Welch. *Asymptotiquement, il n'existe pas de code linéaire binaire $[n, \rho n, d]$ pour $d \geq \epsilon n$ dès que*

$$\rho < h\left(\frac{1}{2} - \sqrt{\epsilon \bar{\epsilon}}\right).$$

Fig. 4. Bornes de Hamming et de V-G

Fig. 5. Bornes de McEliece-Rodemich-Rumsey-Welch et de V-G

10 NP-complétude de problèmes sur les codes

10.1 NP-complet ou NP-ardu?

Un problème X est *NP-ardu* si tout problème dans la classe NP se retrouve dans la classe P dès que X est dans P. Un problème X est *NP-complet* s'il est dans NP en plus d'être *NP-ardu*. Pour montrer un tel résultat, il faut d'abord donner un algorithme non-déterministe polynomial pour X puis montrer comment transformer les exemplaires de n'importe quel problème Y de NP en un exemplaire de X en temps polynomial. Bien sur, il est difficile de montrer une telle chose pour tous les problèmes de NP. Néanmoins, Cook a montré que le problème SAT est NP-complet. À partir d'un premier problème NP-ardu A on peut montrer qu'un autre problème B l'est en donnant une transformation polynomial des exemplaires du problème A aux exemplaires du problème B. Si B est dans P alors A l'est aussi, ainsi que tous les problèmes dans NP. On connait déja de nombreux problèmes NP-complets. Un exemple courant de problème NP-complet est 3DM:

Definition 35 Matching Tri-Dimensionel (3DM). Exemplaire:
DONNÉE: Un sous-ensemble $U \subseteq T \times T \times T$ pour un ensemble fini T.
PROBLÈME: Existe-t-il un sous-ensemble W de U de taille $|T|$ tel que pour tout $(w_1, w_2, w_3), (w'_1, w'_2, w'_3) \in W$ on a $w_1 \neq w'_1, w_2 \neq w'_2, w_3 \neq w'_3$?

La réduction des exemplaires de SAT à ceux de 3DM est un résultat classique de NP-complétude. On peut le trouver dans le livre de Garey et Johnson. Afin de montrer la NP-complétude de certains problèmes sur les codes on va montrer comment transformer les exemplaires de 3DM en ceux des problèmes considérés.

10.2 correction d'erreurs

On montre dans cette section que le problème de la correction d'erreurs pour un code linéaire quelconque est NP-ardu. On défini d'abord formellement ce problème.

Notations: Pour une matrice binaire $\mathbf{H}$ et un mot binaire $\mathbf{w}$ on défini la distance au code engendreé par $\mathbf{H}$ par $\delta(\mathbf{w}) = \min\{\Delta(\mathbf{w}, \mathbf{c}) : \mathbf{w} \neq \mathbf{c}, \mathbf{c}\mathbf{H}^\top = \mathbf{0}\}$. Rappelons que le poids $w(\mathbf{x})$ d'un mot binaire est le nombre de position non-nulles, c'est à dire $w(\mathbf{x}) = \Delta(\mathbf{x}, \mathbf{0})$.

Definition 36 plus proche mot de code (PPMC). Exemplaire:
DONNÉE: Une matrice binaire $\mathbf{H}$, un mot binaire $\mathbf{w}$.
PROBLÈME: Trouver $\mathbf{c}$ tel que $\mathbf{c}\mathbf{H}^\top = \mathbf{0}$ et $\Delta(\mathbf{w}, \mathbf{c}) = \delta(\mathbf{w})$.

Il est évident qu'un algorithme polynomial au problème PPMC nous donne également une solution au problème DC:

Definition 37 distance au code (DC). Exemplaire:
DONNÉE: Une matrice binaire $\mathbf{H}$, un mot binaire $\mathbf{w}$.
PROBLÈME: Trouver $\delta(\mathbf{w})$.

D'autre part, le problème DC est polynomialement équivalent au problème suivant:

Definition 38 poids minimal d'un coset (PMCS). Exemplaire:
DONNÉE: Une matrice binaire $\mathbf{H}$, un mot binaire $\mathbf{y}$, un entier k.
PROBLÈME: Existe-t-il un $\mathbf{x} \neq \mathbf{0}$ tel que $\mathbf{x}\mathbf{H}^\top = \mathbf{y}$ et $w(\mathbf{x}) \leq k$?

Pour voir cette équivalence, il suffit de comprendre les deux réductions suivantes. On nomme DC_{P} et $PMCS_{\mathrm{P}}$ des algorithmes hypothétiques polinomiaux pour résoudre DC et PMCS.

Algorithme 10.1 ($PMCS(\mathbf{H}, \mathbf{y}, k)$ **)**

1: trouver un $\mathbf{x}$ tel que $\mathbf{x}\mathbf{H}^\top = \mathbf{y}$,
2: IF $DC_{\mathrm{P}}(\mathbf{H}, \mathbf{x}) \leq k$ **THEN RETURN** "oui" **ELSE RETURN** "non".

Algorithme 10.2 ($DC(\mathbf{H}, \mathbf{w})$)

1: $k \leftarrow 0$,
2: $\mathbf{y} \leftarrow \mathbf{w}\mathbf{H}^\top$,
3: **WHILE NOT** $PMCS_{\mathrm{P}}(\mathbf{H}, \mathbf{y}, k)$ **DO** $k \leftarrow k+1$,
4: **RETURN** k.

On démontre en fait que PMCS est NP-Complet. Il en découle que DC et PPMC sont NP-ardu.

10.3 NP-complétude de PMCS

On montre d'abord que PMCS est dans NP grace à l'algorithme suivant:

Algorithme 10.3 ($PMCS_{\mathbf{NP}}(\mathbf{H}, \mathbf{y}, k)$)

1: deviner un $\mathbf{x}$,
2: **IF** $\mathbf{x}\mathbf{H}^\top = \mathbf{y}$ et $w(\mathbf{x}) \leq k$ **THEN RETURN** “oui”.

On donne ensuite une réduction de 3DM à PMCS. Soit $T = \{t_1, t_2, ..., t_n\}$. Pour un sous ensemble $U = \{(u_1^1, u_1^2, u_1^3), (u_2^1, u_2^2, u_2^3), ..., (u_m^1, u_m^2, u_m^3)\}$ de $T \times T \times T$ on définie une matrice binaire $\mathbf{H}$ de dimension $m \times 3n$ de la façon suivante:

$$h_{ij} = \begin{cases} 1 \text{ si } u_i^1 = t_j \\ 0 \text{ sinon} \end{cases}, \quad h_{i(j+n)} = \begin{cases} 1 \text{ si } u_i^2 = t_j \\ 0 \text{ sinon} \end{cases}, \quad h_{i(j+2n)} = \begin{cases} 1 \text{ si } u_i^3 = t_j \\ 0 \text{ sinon} \end{cases}.$$

Il suffit maintenant de remarquer que U possède un “matching” de $T \times T \times T$ exactement s'il existe un $\mathbf{x}$ de poids n tel que $\mathbf{x}\mathbf{H}^\top = \mathbf{111...1}$. Puisque tout $\mathbf{x}$ tel que $\mathbf{x}\mathbf{H}^\top = \mathbf{111...1}$ doit avoir poids au moins n par construction de $\mathbf{H}$, il suffit donc de savoir le résultat de $PMCS_{\mathrm{P}}(\mathbf{H}, \mathbf{111...1}, n)$.

10.4 poids des mots de code

Le problème le plus naturel à considérer est celui de trouver un mot de code de poids minimal ou simplement de déterminer le poids minimal des mots de code non-nuls. Ces problèmes sont PPCM et DC restreint aux mots du code. L'an dernier Moshe Vardy a montré que ces versions sont NP-ardu. Il en va donc de même pour PMCS restreint aux mots du code:

Definition 39 poids minimal du coset nul (PMCS0). Exemplaire:
Donnée: Une matrice binaire $\mathbf{H}$, un entier k.
Problème: Existe-t-il un $\mathbf{c} \neq \mathbf{0}$ tel que $\mathbf{c}\mathbf{H}^\top = \mathbf{0}$ et $w(\mathbf{c}) \leq k$?

Nous verrons plus tard la preuve que PMCS0 est NP-complet car elle demande passablement de techniques de codage. On va pour le moment se contenter de montrer qu'un autre problème sur les codes qui est prèsqu'identique est NP-complet. Il s'agit de dire s'il existe ou non des mots de code d'un certain poids:

Definition 40 poids d'un code (PC). Exemplaire:
Donnée: Une matrice binaire $\mathbf{H}$, un entier k.
Problème: Existe-t-il un $\mathbf{c} \neq \mathbf{0}$ tel que $\mathbf{c}\mathbf{H}^\top = \mathbf{0}$ et $w(\mathbf{c}) = k$?

10.5 NP-complétude de PC

On montre d'abord que PC est dans NP grace à l'algorithme suivant:

Algorithme 10.4 ($PC_{\mathbf{NP}}(\mathbf{H}, k)$)

1: deviner un $\mathbf{c}$,
2: IF $\mathbf{cH}^{\top} = \mathbf{0}$ et $w(\mathbf{c}) = k$ **THEN RETURN** "oui".

On donne ensuite une réduction de 3DM à PC. Soit $\mathbf{H}$ la matrice $m \times 3n$ construite à la section 10.3. On construit une nouvelle matrice $\mathbf{M}$ de dimensions $(3nm + 3n + m) \times 3n(m+1)$ à partir de $\mathbf{H}$ de la façon suivante:

$$\mathbf{M} = \left[\begin{array}{c} \mathbf{H}\ \overbrace{\mathbf{I}_m\ \mathbf{I}_m\ \ldots\ \mathbf{I}_m}^{3n} \\ \hline \\ \mathbf{I}_{3n(m+1)} \end{array}\right]$$

Il suffit maintenant de remarquer que U possède un "matching" de $T \times T \times T$ exactement s'il existe un $\mathbf{c}$ de poids $3n^2 + 4n$ tel que $\mathbf{cM}^{\top} = \mathbf{0}$. Il suffit donc de savoir le résultat de $PC_{\mathrm{P}}(\mathbf{M}, 3n^2 + 4n)$, où PC_{P} est un algorithme hypothétique polynomial pour résoudre PC.

10.6 Équivalence de codes

D'autre part, il est peu probable que le problème suivant soit NP-complet, bien qu'il est peu probable qu'il soit dans P:

Definition 41 équivalence de codes (ÉC). Exemplaire:
  DONNÉE: Deux matrices binaires $\mathbf{H_0}, \mathbf{H_1}$.
  PROBLÈME: Les matrices $\mathbf{H_0}$ et $\mathbf{H_1}$ sont-elles des matrices de parité de codes équivalents?

11 Codes cycliques

Les codes cycliques sont des codes linéaires tels que si le mot de code $(c_1, c_2, ..., c_n)$ est dans le code, alors le mot $(c_n, c_1, ..., c_{n-1})$ l'est aussi.

On s'intéresse aux codes cycliques à cause de leurs propriétés algébriques. On va donc définir formellement les codes cycliques comme des objets algébriques.

Soit $\mathcal{F}_q[x]$ l'ensemble des polynomes de la variable x à coéfficients dans $\mathcal{F}_q$. Soit $R_n = \mathcal{F}_q[x]/(x^n - 1)$ l'anneau des polynomes à coéfficients dans $\mathcal{F}_q$ modulo $(x^n - 1)$.

Definition 42. Un *idéal* C de R_n est un sous-espace linéaire de R_n tel que pour tout $c(x) \in C$ et $r(x) \in R_n$, on a $c(x)r(x) \in C$.

ou de façon équivalente

Definition 43. Un *idéal* C de R_n est un sous-espace linéaire de R_n tel que pour tout $c(x) \in C$, on a $xc(x) \in C$.

Definition 44. Un *code cyclique* C est un idéal de R_n.

11.1 polynome générateur

Pour tout code cyclique C il existe un polynome *générateur* g tel que $C = \langle g \rangle$, c'est-à-dire que $C = \{g(x)r(x) : r(x) \in R_n\}$.

Theorem 45. *Soit C un code cyclique de R_n.*

- *il existe un unique polynome $g(x)$ unitaire de degré minimal dans C,*
- *ce polynome $g(x)$ est un générateur de C,*
- *$g(x)$ est un facteur de $x^n - 1$,*
- *tout $c(x) \in C$ peut s'écrire comme $c(x) = f(x)g(x)$ avec $deg(f) \leq n - deg(g)$,*
- *si on pose $g(x) = g_0 + g_1 x + ... + g_r x^r$ une matrice génératrice de C est*

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & \dots & g_r & 0 & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{r-1} & g_r & 0 & \dots & 0 \\ 0 & 0 & g_0 & \dots & g_{r-2} & g_{r-1} & g_r & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & g_0 & g_1 & g_2 & \dots & g_r \end{bmatrix}$$

11.2 polynome de parité

Pour tout code cyclique C il existe un polynome *de parité* h tel que $h(x) = (x^n - 1)/g(x)$, c'est-à-dire que $C = \{c(x) : c(x)h(x) \equiv 0 \pmod{x^n - 1}\}$.

Theorem 46. *si on pose $h(x) = h_0 + h_1 x + ... + h_k x^k$ une matrice de parité de C est*

$$\mathbf{H} = \begin{bmatrix} 0 & \dots & 0 & 0 & h_k & \dots & h_2 & h_1 & h_0 \\ 0 & \dots & 0 & h_k & h_{k-1} & \dots & h_1 & h_0 & 0 \\ 0 & \dots & h_k & h_{k-1} & h_{k-1} & \dots & h_0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ h_k & \dots & h_2 & h_1 & h_0 & \dots & 0 & 0 & 0 \end{bmatrix}$$

11.3 code dual

Le code dual à un code cyclique C est également un code cyclique. Le polynome générateur $g^\perp$ du code dual $C^\perp$ est $g^\perp = x^{deg(h)} h(1/x)$.

11.4 facteurs de $x^n - 1$

Soit $\mathcal{F}_q$ un corps fini quelconque et soit n relativement premier à q. Soit m le plus petit entier tel que $q^m \equiv 1 \pmod{n}$. Alors sur $\mathcal{F}_{q^m}$ le polynome $x^n - 1$ est complètement factorisable sous la forme $x^n - 1 = \prod_{i=0}^{n-1}(x - \alpha^i)$ pour α une racine $n^{\text{ième}}$ primitive de l'unité. En général, si γ est un élément primitif de $\mathcal{F}_{q^m}$ alors $\alpha = \gamma^{(q^m-1)/n}$ est une racine $n^{\text{ième}}$ primitive de l'unité.

Puisque $g(x)h(x) = x^n - 1$, il existe $K \subseteq \{0, 1, 2, ..., n-1\}$ tel que

$$g(x) = \prod_{i \in K}(x - \alpha^i) \text{ et } h(x) = \prod_{i \notin K}(x - \alpha^i).$$

11.5 cosets cyclotomiques et polynômes minimaux

On définit sur $\mathcal{F}_q$, le coset cyclotomique de $s \pmod{n}$ comme

$$C_s = \{s, sq, sq^2, \ldots, sq^{m_s-1}\}$$

où m_s est le plus petit entier tel que $sq^{m_s} \equiv s \pmod{n}$.

Notons que $m = m_1$ (cardinalité de C_1) nous donne le plus petit corps $\mathcal{F}_{q^m}$ où $x^n - 1$ possède des racines.

Le polynôme minimal de α^s est

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

il est le polynôme de $\mathcal{F}_q[x]$ ayant degré minimal parmis tous ceux qui ont α^s comme racine. Il satisfait six propriétés importantes:

(M1) $M^{(s)}(x)$ est irréductible sur $\mathcal{F}_q$.
(M2) si $f(x)$ est un polynôme de $\mathcal{F}_q[x]$ tel que $f(\alpha^s) = 0$ alors $M^{(s)}(x) | f(x)$.
(M3) $M^{(s)}(x) | x^{n+1} - x$.
(M4) $\deg(M^{(s)}(x)) \leq m$.
(M5) le polyôme minimal d'une racine primitive a degré m.
(M6) α^s et α^{sq} ont le même polynôme minimal.

Exemple, code de Hamming: le code de Hamming H_m a pour matrice de parité

$$\mathbf{H} = \left[\, 1 \ \alpha \ \alpha^2 \ ... \ \alpha^{2^m-1} \,\right]$$

où α est une élément primitif de $\mathcal{F}_{2^m}$. $c(x)$ est donc un mot de code si et seulement si $c(\alpha) = 0$. Par la propriété (M2), ceci implique que $c(x)$ est un multiple de $M^{(1)}(x)$ et donc que le code est engendré par le polynôme $g(x) = M^{(1)}(x)$ et a pour matrice génératrice:

$$\mathbf{G} = \begin{bmatrix} M^{(1)}(x) & & & \ldots & 0 & 0 \\ 0 & M^{(1)}(x) & & & \ldots & 0 \\ 0 & 0 & M^{(1)}(x) & & & \\ ... & ... & ... & ... & ... & \\ 0 & 0 & 0 & ... & M^{(1)}(x) & \end{bmatrix}$$

Exemple, BCH corrigeant 2 erreurs: le code de BCH corrigeant 2 erreurs a pour matrice de parité

$$\mathbf{H} = \begin{bmatrix} 1 \ \alpha \ \alpha^2 \ ... \ \alpha^{2^m-1} \\ 1 \ \alpha^3 \ \alpha^6 \ ... \ \alpha^{3(2^m-1)} \end{bmatrix}.$$

$c(x)$ est donc un mot de code si et seulement si $c(\alpha) = c(\alpha^3) = 0$. On conclut donc comme ci-haut qu'il est engendré par le polynôme $g(x) = M^{(1)}(x)M^{(3)}(x)$.

11.6 borne de BCH

Theorem 47. *Soit C un code cyclique généré par le polynome $g(x)$ tel qu'il existe des entiers $b \geq 0, \delta \geq 1$ et tel que*

$$g(\alpha^b) = g(\alpha^{b+1}) = ... = g(\alpha^{b+\delta-2}) = 0.$$

Alors la distance minimale de C est d'au moins δ.

12 Codes BCH

Sur un corps fini $\mathcal{F}_q$, on définit un code BCH de distance de design δ pour un entier $b \geq 0$ à partir du polynome générateur

$$g(x) = \text{ppcm}\left\{M^{(b)}(x), M^{(b+1)}(x), ..., M^{(b+\delta-2)}(x)\right\}.$$

Theorem 48. *Un code BCH de longueur n et de distance de design δ a une distance minimale $d \geq \delta$ et une dimension $k = n - deg(g) \geq n - m(\delta - 1)$.*

Remarques

- les codes BCH *au sens restreint* sont le cas $b = 1$,
- les codes BCH *primitifs* sont les cas $n = q^m - 1$,
- pour b fix, les codes BCH sont *emboités* c'est-à-dire que le BCH de distance de design δ_1 est un sous-ensemble du BCH de distance de design δ_2 si $\delta_2 \leq \delta_1$,
- en général le dual d'un code BCH n'est pas un code BCH.

Exemple:

δ	g	$i : g(\alpha^i) = 0$	k	d
1	1	$-$	15	1
3	$M^{(1)}$	$1, 2, 4, 8$	11	3
5	$M^{(1)}M^{(3)}$	$1-4, 6, 8, 9, 12$	7	5
7	$M^{(1)}M^{(3)}M^{(5)}$	$1-6, 8-10, 12$	5	7
$9, 11, 13, 15$	$M^{(1)}M^{(3)}M^{(5)}M^{(7)}$	$1-14$	1	15

Fig. 6. codes BCH de longueur 15.

δ	g	k	d
1	1	23	1
$3, 5$	$M^{(1)}$	12	7
$7-23$	$M^{(1)}M^{(5)}$	1	23

Fig. 7. codes BCH de longueur 23.

δ	g	k	d
1	1	31	1
3	$M^{(1)}$	26	3
5	$M^{(1)}M^{(3)}$	21	5
7	$M^{(1)}M^{(3)}M^{(5)}$	16	7
$9, 11$	$M^{(1)}M^{(3)}M^{(5)}M^{(7)}$	11	11
$13, 15$	$M^{(1)}M^{(3)}M^{(5)}M^{(7)}M^{(11)}$	6	15
$17-31$	$M^{(1)}M^{(3)}M^{(5)}M^{(7)}M^{(11)}M^{(15)}$	1	31

Fig. 8. codes BCH de longueur 31.

12.1 Distance minimale effective

Les quelques théoremes suivants nous donne une idée de la relation entre la distance de design d'un code BCH (δ) et sa distance minimale effective (d).

Theorem 49. *Un code BCH de longueur $n = q^m - 1$ et de distance de design $\delta = q^h - 1$ sur $\mathcal{F}_q$ a sa distance minimale effective $d = \delta$.*

Theorem 50. *Un code BCH de longueur $n = q^m - 1$ et de distance de design δ sur $\mathcal{F}_q$ a sa distance minimale effective $d \leq q\delta - 1$.*

12.2 Symboles d'information

Soit $I(n, \delta)$ le nombre de symboles d'information dans un mot de code d'un code BCH primitif au sens restreint. Bien sur $I(n, \delta) = n - deg(g(x))$ où $g(x)$ est le polynome générateur du code. Néanmoins il est est préférable de trouver une borne générique sur $I(n, \delta)$ sans tenir compte du $g(x)$ précis. C'est ce que fait le théoreme suivant.

Theorem 51. *Le nombre de symboles d'information d'un code BCH au sens restreint de longueur $n = q^m - 1$ et de distance de design $\delta = q^\lambda$ sur $\mathcal{F}_q$ est*

$$I(q^m - 1, q^\lambda) \leq \sum_{i=0}^{r} a_i \rho_i^m$$

pour $r = m - \lambda$ et des constantes $a_1, ..., a_r, \rho_1, ..., \rho_r$ qui ne sont fonctions que de r et telles que $|\rho_i| < q$.

12.3 Les codes BCH sont mauvais

Une famille de code $\{C_i\}$ de paramètres $[n_i, k_i, d_i]$ est *bonne* si

$$\lim_{i \to \infty} k_i/n_i = R > 0 \text{ et } \lim_{i \to \infty} d_i/n_i = D > 0.$$

Sinon on dit de la famille qu'elle est *mauvaise.*

Theorem 52. *Toute famille de codes BCH primitifs restreint est mauvaise.*

Theorem 53 Berlekamp. *Toute famille de codes BCH primitifs binaire de taux R possède assymptotiquement ($n \to \infty$) une distance minimale*

$$d \sim 2n \frac{-\log_e R}{\lg n}.$$

13 Codes Reed-Solomon

13.1 definition

Un code Reed-Solomon (RS) sur $\mathcal{F}_q$ est un code BCH de longueur $N = q - 1$. Le polynome générateur du code RS de distance de design δ est

$$g(x) = M^{(b)}(x)M^{(b+1)}(x)\cdots M^{(b+\delta-2)}(x) = (x-\alpha^b)(x-\alpha^{b+1})\cdots(x-\alpha^{b+\delta-2}).$$

La dimension d'un tel code est $K = N - deg(g) = N - \delta + 1$ et sa distance minimale est $D \geq \delta$. Mais puisque d'autre part par la borne de Singleton on a que $D \leq N - K + 1$ on a donc

$$D = \delta = N - K + 1.$$

Theorem 54 RS étendus. *Soit C un code RS de paramètres $[N = q-1, K, D]$ avec polynôme générateur*

$$g(x) = (x-\alpha)(x-\alpha^2)\cdots(x-\alpha^{D-1}).$$

Le code $C^+ = \{(c_0c_1c_2...c_{N-1}) : (c_1c_2...c_{N-1}) \in C, c_0 = -\sum_{i=1}^{N-1} c_i\}$ est un code $[N+1, K, D+1]$.

13.2 codage

Theorem 55 interpolation de Lagrange. *Soit $\alpha^{i_1}\alpha^{i_2}...\alpha^{i_K}$ des éléments distincts de $\mathcal{F}_q$ pour $K < q$ et $\beta_1\beta_2...\beta_K$ des éléments quelconques de $\mathcal{F}_q$. Il existe un unique polynôme $p(x)$ sur $\mathcal{F}_q$ tel que $deg(p) < K$ et que pour $1 \leq j < K$*

$$g(\alpha^{i_j}) = \beta_i.$$

Preuve: considérons le système linéaire suivant:

$$\begin{bmatrix} 1 & \alpha^{i_1} & \alpha^{2i_1} & \cdots & \alpha^{(K-1)i_1} \\ 1 & \alpha^{i_2} & \alpha^{2i_2} & \cdots & \alpha^{(K-1)i_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{i_K} & \alpha^{2i_K} & \cdots & \alpha^{(K-1)i_K} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{K-1} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{bmatrix}$$

puisque les α^{i_j} sont distincts, cette matrice est Vandermonde et donc elle est inversible. Il existe donc une et une seule solution $(p_0p_1...p_{K-1})$ pour chaque $(\beta_1\beta_2...\beta_K)$.

Theorem 56. *Consiérons le code définit ainsi*

$$C = \{(p(1)p(\alpha)...p(\alpha^{N-1})) : p \text{ est un polynôme de degré} < K \text{ sur } \mathcal{F}_q\}.$$

Le code C est un RS code $[N = q-1, K, N-K+1]$.

Preuve: d'abord il est clair que la dimension de C est K car il existe q^K valeurs de p et $p \neq p'$ implique que $(p(1)p(\alpha)...p(\alpha^{N-1})) \neq (p'(1)p'(\alpha)...p'(\alpha^{N-1}))$. Soient p et p' deux polynômes tels que

$$\Delta\left((p(1)p(\alpha)...p(\alpha^{N-1}), (p'(1)p'(\alpha)...p'(\alpha^{N-1})\right) < N - K + 1.$$

Ceci implique qu'ils existent $i_1, i_2, ..., i_K$ tels que $p(\alpha^{i_j}) = p'(\alpha^{i_j})$. Par le théoreme précédent ceci entraîne que $p = p'$. Donc il n'existe pas de pair c, c' de mots distincts tels que $\Delta(c, c') < N - K + 1$.

13.3 codes binaires

On peut construire à partir d'un code RS C de paramètres $[N, K, D]$ sur $\mathcal{F}_{2^m}$ un code binaire $C^\#$ de paramètres $[n = N, K \geq k \geq n - m(N-K), d \geq D]$ en considérant uniquement les mots de C ayant que des composantes dans $\mathcal{F}_2$.

On peut également construire à partir d'un code RS C de paramètres $[N, K, D]$ sur $\mathcal{F}_{2^m}$ un code binaire $C^\rightarrow$ de paramètres $[n = Nm, k = Km, d \geq D]$ en remplaçant dans les mots de code de C chaque symbole de $\mathcal{F}_{2^m}$ par m bits. Pour obtenir ces m bits il faut spécifier une base $b_1, b_2, ..., b_m$ de $\mathcal{F}_{2^m}$ sur $\mathcal{F}_2$.

On peut enfin construire à partir d'un code RS C de paramètres $[N, K, D]$ sur $\mathcal{F}_{2^m}$ un code binaire $C^\uparrow$ de paramètres $[n = N(m+1), k = Km, d \geq 2D]$ en remplaçant dans les mots de code de C chaque symbole de $\mathcal{F}_{2^m}$ par m bits et en ajoutant un bit de parité à chaque bloc de m bits.

13.4 les codes RS "binarisés" sont mauvais

Les codes binaires $C^{\rightarrow}$ et $C^{\uparrow}$ ne forment pas de famille de codes assymptotiquement bonne au sens déjà vu.

13.5 correction de paquets d'erreurs

Quand les erreurs se présentent par paquets plutot que de façon complètement indépendantes la structure de blocs obtenue en formant un code binaire d'un Reed-Solomon est propice à leur correction. Si le Reed-Solomon original a distance minimale D sur $\mathcal{F}_{2^m}$ alors le code binaire obtenu pourra corriger $D/2$ blocs erronnés. Un paquet de b bits erronnés pourra affecter r symboles sur $\mathcal{F}_{2^m}$ pour

$$(r-2)m+2 \leq b \leq (r-1)m+1.$$

De nombreux paquets peuvent être corrigés si r est beaucoup plus petit que D.

14 Codes Concaténés et Justesen

14.1 concaténation de codes

Un code concaténé C est obtenu à partir de deux codes C_E dit *externe* sur $\mathcal{F}_{q^m}$ de paramètres $[n_E, k_E, d_E]$ et C_I dit *interne* sur $\mathcal{F}_q$ de paramètres $[n_I, k_I = m, d_I]$. Le code ainsi obtenu sera un code sur $\mathcal{F}_q$ de paramètres $[n = n_E n_I, k = k_E k_I = mk_E, d \geq d_E d_I]$.

14.2 code interne aléatoire ou maximal

Si l'on construit des codes concaténés en prenant comme code externe un Reed-Solomon $[2^m, R2^m, (1-R)2^m+1]$ et comme code binaire interne $[\sigma m, m, \rho m]$ un code près de la borne de V-G $(1/\sigma \approx 1-h(\rho/\sigma))$ on obtiendra des codes concaténés de paramètres $[\sigma m 2^m, Rm2^m, (1-R)\rho m 2^m]$. En posant $N = \sigma m 2^m$, $r = \frac{1}{\sigma}$ et en éliminant ρ on obtient

$$[N, RrN, (1-R)h^{-1}(1-r)N].$$

Si l'on fixe le produit $\gamma = Rr$ il faudra prendre le r qui maximise l'expression suivante $(1-\frac{\gamma}{r})h^{-1}(1-r)$ pour obtenir les meilleurs codes de ce type. La figure 9 nous montre le graphe de cette fonction comparé à la courbe de V-G.

On peut trouver les codes internes en prenant des matrices aléatoires de taille $\sigma m \times m$ jsuqu'à ce que l'on en trouve une dont la distance minimale associée soit assez près de la borne de V-G.

14.3 codes multiplicatifs

Considérez les codes binaires $C_\alpha = \{[u, \alpha u] : u \in \mathcal{F}_{2^m}\}$ pour $\alpha \in \mathcal{F}_{2^m} - \mathbf{0}$. Pour $\alpha \neq \beta$ on a $C_\alpha \bigcap C_\beta = \{\mathbf{0}\}$ car un mot $[u, v] \neq \mathbf{0}$ appartient seulement à $C_{(vu^{-1})}$.

14.4 les Codes multiplicatifs sont bons

Les codes de type C_α forment une bonne famille de codes. Pour voir ceci il suffit de remarquer d'abord qu'ils ont toujours un taux de transmission $R = 1/2$, puis de compter combien de codes peuvent avoir une distance minimale faible.

Il y a $2^m - 1$ codes de type C_α au total. Il y a $\sum_{i=1}^{\lambda 2m} \binom{2m}{i} \leq 2^{h(\lambda)2m}$ mot de longueur $2m$ de poids inféfieur ou égal à $\lambda 2m$. Puisque chaque mot ne peut appartenir à plus d'un C_α, dès que ce nombre $2^{h(\lambda)2m}$ est inférieur à $2^m - 1$, il doit exister des codes C_α de distance minimale supérieure à $\lambda 2m$. Ceci sera le cas quand $\lambda < h^{-1}(1/2) \approx 0.11$.

Fig. 9. Bornes de Zyablov et de V-G

14.5 codes de Justesen

L'idée de Justesen fut de concaténer les codes de type C_α avec un Reed-Solomon $[2^m-1, R(2^m-1), (1-R)(2^m-1)+1]$. Justesen suggère d'utiliser comme codes internes tous les C_α. Selon moi, il est aussi efficace et plus simple de prendre le meilleur des C_α. Il suffit pour cela de les énumérer tous et de prendre celui qui a la plus grande distance minimale. On sais que cette distance minimale sera au moins $h^{-1}(1/2)2m$. Le code concaténé aura donc des paramètres

$$[N, R'N, (1-2R')h^{-1}(1/2)N]$$

pour $N = 2m(2^m-1)$ et $R' = R/2$. Ces codes sont sur la droite passant par les points $(0, h^{-1}(1/2))$ et $(1/2, 0)$ et tangent au point $(0.3, 0.4h^{-1}(1/2))$ de la courbe de Zyablov.

14.6 décodage des codes concaténés

décodage naïf Consulter la thèse de Sendrier.

décodage de Block-Zyablov Consulter la thèse de Sendrier.

14.7 codes de Forney

Il est assez simple de constater que les codes concaténés contiennent des codes près de la borne de Shannon qui peuvent être construits, codés et décodés en temps polynomial. La construction utilise le théorème de Shannon que revoici.

Theorem 57. *Soit ϵ, tel que $0 < \epsilon < 1$ et soit R tel que $0 < R < \mathbf{C}(BS_\epsilon)$. Il existe un entier n_0, une constante positive $\alpha < 1$ et une famille infinie de codes $\{C_n\}_{n>n_0}$ de paramètres (n, M_n, d_n) tels que*

$$M_n = \lceil 2^{Rn} \rceil \quad et \quad \hat{e}(C_n) < \alpha^n$$

pour tout $n > n_0$.

De plus, pour n_0 assez grand, un code aléatoire aura de tels paramètres sauf avec probabilité inférieure à 1/2.

Soit $\epsilon > 0$ une probabilité d'erreur. Soit $\delta > 0$ une costante positive. Soit $q = 2^m$ pour un m assez grand. Soit R tel que $0 < R < \mathbf{C}(BS_\epsilon)$. Considérons un code aléatoire binaire de longueur m contenant $M_m = \lceil 2^{Rm} \rceil$ mots. Pour m assez grand, la probabilité d'erreur de décodage moyenne du code, $\hat{e}(C_m) < \alpha^m$.

Si l'on concatène ce code avec un Reed-Solomon $[2^m, 2^m(1-\delta), 2^m\delta+1]$, il en résulte un code de longueur $N = m2^m$ ayant un taux de transmission au moins $R(1-\delta)$. De plus, par la loi des grands nombres, la probabilité qu'une fraction $\delta/2$ des 2^m blocs soient en erreur alors qu'une fraction α^m est espérée, est exponentiellement faible en 2^m, disons inférieure à β^{2^m} pour une constante positive $\beta < 1$. Donc le décodage successif des deux niveaux échouera seulement avec probabilité inférieure à β^{2^m}. Ceci n'est pas tout à fait exponentiellement faible en N, mais néanmoins, β^{2^m} est plus petit que n'importe quel inverse de polynôme en N:

Theorem 58. *Por tout entier positif k et tout m assez grand*

$$\beta^{2^m} < 1/N^k.$$

Ceci découle du fait que $N = m2^m$ et donc que $\beta^{2^m} = N^{-\Theta(2^m/(m+\lg m))}$. L'exposant $2^m/(m+\lg m)$ ne peut être borné par aucune constante.

15 Codes Reed-Solomon généralisés

Un code Reed-Solomon Généralisé (RSG) sur $\mathcal{F}_{q^m}^N$ est caractérisé par deux vecteurs $\boldsymbol{\alpha}, \mathbf{v}$ de longueur N avec $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_N) \in \mathcal{F}_{q^m}^N$ pour des α_i distincts et $\mathbf{v} = (v_1, v_2, ..., v_N) \in (\mathcal{F}_{q^m} - \{\mathbf{0}\})^N$. Les mots d'un tel code sont obtenus à partir des polynômes F de degré $< K$ par

$$c = (c_1 c_2 ... c_N) = (v_1 F(\alpha_1), v_2 F(\alpha_2), ..., v_N F(\alpha_N)).$$

15.1 le dual d'un code RSG est RSG

À chaque $RSG(\boldsymbol{\alpha}, \mathbf{v})$ de dimension K on peut associer un autre $RSG(\boldsymbol{\alpha}, \mathbf{v'})$ de dimension $N-K$ dual au premier. Ceci veut dire que la matrice de parité d'un code $RSG(\boldsymbol{\alpha}, \mathbf{v})$ est donnée par

$$\mathbf{H} = \begin{bmatrix} v'_1 & v'_2 & \dots & v'_N \\ v'_1\alpha_1 & v'_2\alpha_2 & \dots & v'_N\alpha_N \\ \vdots & \vdots & \ddots & \vdots \\ v'_1\alpha_1^{N-K-1} & v'_2\alpha_2^{N-K-1} & \dots & v'_N\alpha_N^{N-K-1} \end{bmatrix}.$$

16 Codes Alternants

Un code alternant $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{v'})$ est un code sur $\mathcal{F}_q$ qui est obtenu du $RSG(\boldsymbol{\alpha}, \mathbf{v})$ en conservant les mots ayant que des composantes dans $\mathcal{F}_q$. La matrice $\mathbf{H}$ est aussi la matrice de parité du code $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{v'})$ sur $\mathcal{F}_q$.

16.1 décodage des codes Alternants: algorithme de Berlekamp

Soit C un code alternant $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{v'})$ avec matrice de parité $\mathbf{H}$. Soit $r = N - K - 1$. Supposons que le mot $\mathbf{c}$ soit transmis et qu'un vecteur d'erreur $\mathbf{e}$ de poids $t < r/2$ lui soit ajouté. Appelons $\mathbf{w} = \mathbf{c} \oplus \mathbf{e}$ le mot reçu. Soient $X_1 = \alpha_{i_1}, X_2 = \alpha_{i_2}, \ldots, X_t = \alpha_{i_t}$ les t positions non-nulles de $\mathbf{e}$, et soient $Y_1 = e_{i_1}, Y_2 = e_{i_2}, \ldots, Y_t = e_{i_t}$ les valeurs des erreurs en ces t positions.

On pose de façon formelle, le polynôme localisateur

$$\sigma(z) = \prod_{i=1}^{t} (1 - X_i z)$$

ainsi que le polynôme évaluateur

$$\omega(z) = \sum_{j=1}^{t} Y_j v'_{i_j} \prod_{k=1, k\neq j}^{t} (1 - X_k z)$$

Theorem 59.

$$Y_j = \frac{\omega(X_j^{-1})}{v'_{i_j} \prod_{k=1, k\neq j}^{t} (1 - X_k X_j^{-1})}$$

Theorem 60.

$$\omega(x) \equiv \sigma(x) S(x) \pmod{x^r}$$

Algorithme 16.1 (Berlekamp $decode(\mathbf{H}, \mathbf{w})$ **)**

1: Calculer le syndrome $\mathbf{S} = [s_0, s_1, ..., s_{r-1}] = \mathbf{wH}^\top$,
2: Définir un polynôme $S(x) = s_0 + s_1 x + \cdots + s_{r-1} x^{r-1}$,
3: Appliquer l'algorithme $Euclide(S(x), x^r)$ afin de trouver a_k, b_k, r_k tels que

$$r_k(x) = a_k(x) x^r + b_k(x) S(x) \quad \text{et} \quad deg(r_k) < r/2 \quad \text{mais} \quad deg(r_{k-1}) \geq r/2,$$

4: Poser $\sigma(x) = b_k(x)/b_k(0)$ et $\omega(x) = r_k(x)/b_k(0)$,
5: Trouver les racines de $\sigma(x)$ par énumération.

16.2 les Codes Alternants sont bons

17 Codes Classiques de Goppa

18 Codes Géométriques

19 Graphes et codes

19.1 Graphes Expandeurs

19.2 Codes expandeurs

19.3 Codes superconcentrateurs

20 Applications à la cryptographie

20.1 partages de secrets

20.2 Cryptosystème de McEliece

20.3 réductions entres protocoles

20.4 Winer wire-tap channel

20.5 Extension de Maurer et BBCM

20.6 Protocoles cryptographiques à partir d'un canal bruyant

21 Codes quantiques

IFT6170 Devoir 1

à rendre au début du cours le mercredi 1 octobre 1997

Polynôme Irréductible [4 points]

Considérez l'ensemble des nombres modulo 17 étiquetés ainsi:

$$\{\mathbf{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G}\},$$

les symboles "**A,B,C,D,E,F,G**" repésentant les nombres 10,11,12,13,14,15,16. On utilise sur ces nombres les opérations d'addition, soustraction et multiplication modulo 17. Par exemple $F + 1 \equiv G \pmod{17}$ et $3 * A \equiv D \pmod{17}$. On considère les polynômes sur $\mathcal{F}_{17}$ comme une suite finie de symboles parmis l'ensemble ci-haut. Par exemple (1 0 1 0 0 0 A B C D E F G) correspond à

$$1x^{12} + 1x^{10} + Ax^6 + Bx^5 + Cx^4 + Dx^3 + Ex^2 + Fx + G.$$

Donnez un polynôme **irréductible** sur $\mathcal{F}_{17}$ de **degré 23** contenant votre date de naissance de façon consécutive dans sa représentation ainsi qu'une preuve de ce fait. Par exemple pour 17-02-62

(1 7 0 2 6 2 0 0 0 0 0 0 0 0 0 0 0 0 A 5 4 2 1 F 7)

pourrait être un tel polynôme contenant ma date de naissance. (Je n'ai pas vérifié s'il est irréductible.)

Donnez également un élément primitif du corps fini engendré par votre polynôme. Pour vous aider, sachez que $17^{23} - 1 = 2^4 * 47 * 2655261821922809016297748 1$. Vous pouvez utiliser les moyens (ordinateurs, logiciels, etc) que vous voudrez pour obtenir votre réponse.

Dual des codes de Hamming [4 points]

Les codes de Hamming forment une famille infinie $\{\mathcal{H}_m\}_{m>0}$ de codes linéaires binaires ayant les paramètres $[2^m - 1, 2^m - m - 1, 3]$. Donnez les paramètres $[n_m, k_m, d_m]$ des codes $\{\mathcal{H}_m^\top\}_{m>0}$ duaux des codes de Hamming en fonction de m. Ces codes sont-ils parfaits? Prouvez vos réponses.

Codes ISBN [4+1 points]

Le code ISBN que l'on trouve sur les livres est un code linéaire sur $\mathcal{F}_{11}$ où l'on utilise l'alphabet

$$\{\mathbf{0,1,2,3,4,5,6,7,8,9,X}\},$$

le symbole "**X**" repésentant le nombre 10. Les paramètres de ce code sont $[10, k, d]$. Utilisez un certain nombre de livres (donnez les titres et numéro ISBN des livres utilisés) afin de déterminer une matrice génératrice pour ce code, sous forme systématique. Donnez les valeurs de k et d pour ce code. Démontrez finalement que ce code a la propriété supplémentaire que si l'on permute deux symboles différents d'un mot de code, le mot résultant n'est plus un mot de code. Justifiez vos réponses.

Un point supplémentaire à celui ou celle qui trouvera un plus petit ensemble de livres (je veux les titres et les codes ISBN) dont les codes ISBN sont linéairement dépendants...

Canaux de transmission [total 4 points]

–canal majorité [1.5 points]

Considérez le canal suivant:

$$MAJ_\epsilon^k(x) = \text{maj } \{\overbrace{BS_\epsilon(x), BS_\epsilon(x), ..., BS_\epsilon(x)}^{2k-1}\}.$$

C'est-à-dire que $MAJ_\epsilon^k(x)$ retourne le bit majoritaire parmis $2k - 1$ exécutions indépendantes de $BS_\epsilon(x)$. Trouvez un ϵ_k tel que

$$MAJ_\epsilon^k(x) = BS_{\epsilon_k}(x).$$

Montrez que

$$\lim_{k\to\infty} \epsilon_k = 0.$$

–canal xor [1.5 points]

Considérez le canal suivant:

$$XOR^k_\epsilon(x) = \overbrace{BS_\epsilon(x) \oplus BS_\epsilon(x) \oplus ... \oplus BS_\epsilon(x)}^{2k-1}.$$

C'est-à-dire que $XOR^k_\epsilon(x)$ retourne le ou–exclusif de $2k-1$ exécutions indépendantes de $BS_\epsilon(x)$. Trouvez un ϵ_k tel que

$$XOR^k_\epsilon(x) = BS_{\epsilon_k}(x).$$

Montrez que

$$\lim_{k\to\infty} \epsilon_k = 1/2.$$

–canal mixte [1 point]

Soient $U = BS_\epsilon(x)$ et $V = BS_\epsilon(x)$ deux échantillons indépendants obtenus à travers un canal bruyant de probabilité d'erreur ϵ. Considérez le canal suivant:

$$MIX_\epsilon(x) = \begin{cases} b & \text{si } U = V = b \\ \varepsilon & \text{si } U \neq V \end{cases}.$$

C'est-à-dire que $MIX_\epsilon(x)$ retourne le bit commun obtenu de deux exécutions indépendantes de $BS_\epsilon(x)$, mais la chaîne nulle si ces deux exécutions ne donne pas la même réponse. Trouvez des α, β tels que

$$MIX_\epsilon(x) = EF_\alpha(BS_\beta(x)).$$

IFT6170 Devoir 2

à rendre au début du cours le mercredi 15 octobre 1997

Théorie de l'information [8 points]

Soit X une variable aléatoire prenant comme valeur à probabilité égale n'importe quel mot binaire de n bits. Soit f une quelconque fonction de n bits dans k bits telle que $f(x_1x_2...x_n) = x_{i_1}x_{i_2}...x_{i_k}$ pour une certain liste d'indices $i_1, i_2, ..., i_k$. Soit BS_ϵ^n un canal binaire symétrique d'erreur ϵ. Soit $hash$ la fonction de n bits dans 1 bit telle que $hash(x_1x_2...x_n) = x_1 \oplus x_2 \oplus ... \oplus x_n$.

1. Quelles sont $\mathbf{H}(X), \mathbf{H}(hash(X)), \mathbf{H}(f(X)), \mathbf{H}(BS_\epsilon^n(X))$?
2. Montrez que $\mathbf{H}(X|f(X))$ et $\mathbf{H}(f(X)|X)$ sont les même quelque soit $i_1, i_2, ..., i_k$ et donnez-en les valeurs.
3. Quelles sont $\mathbf{H}(X|BS_\epsilon^n(X))$ et $\mathbf{H}(BS_\epsilon^n(X)|X)$?
4. Quelles sont $\mathbf{H}(hash(X)|f(X))$ et $\mathbf{H}(hash(X)|BS_\epsilon^n(X))$?
5. Prenons un exemple avec $k = 0.99n$ et $\epsilon = 0.49$. Considérez la situation suivante: Alice et Bob connaissent la valeur de X et Eve connait soit la valeur de $f(X)$ pour un f de son choix comme ci-haut, soit celle de $BS_{0.49}^n(X)$. Alice et Bob désirent obtenir un bit secret à partir de X. Pour cela ils utilisent $B \leftarrow hash(X)$. Laquelle des deux stratégies possibles – voir n'importe quel 99% des bits de X ou voir X à travers un canal faisant 49% d'erreurs – donnera le plus d'information à Eve au sujet de B? (Comparez $\mathbf{I}(B; f(X))$ et $\mathbf{I}(B; BS_\epsilon^n(X))$)
6. Est-il possible de trouver une quelconque fonction g de n bits dans 1 bit telle que $\mathbf{I}(g(X); BS_\epsilon^n(X)) = 0$? Démontrez votre réponse.

Canal d'Effacement Erroné [4 points]

Considérons le canal de la figure ci-haut. Supposons qu'en input $P[X = 0] = p_0$ et $P[X = 1] = 1 - p_0$. Calculez l'information mutuelle entre l'input et l'output de ce canal en fonction de p, q, p_0, puis trouvez la capacité de ce canal et dites enfin quelle valeur de p_0 produit la capacité.

PLUS et encore PLUS [4 points]

Démontrez que pour toutes variables aléatoires X et Y on a $\mathbf{H}(X+Y|Y) = \mathbf{H}(X|Y)$ et que si X et Y sont indépendantes alors $\mathbf{H}(X+Y) \geq \max\{\mathbf{H}(X), \mathbf{H}(Y)\}$. Donnez un exemple de variables dépendantes pour lesquelles cette inégalité ne tient plus.

IFT6170 Devoir 3

à rendre au début du cours le mercredi 5 novembre 1997

Équivalence de problèmes sur les codes [4 points]

Démontrez l'équivalence polynomiale des problèmes PMCS0 et PPMC restreints aux mots de codes (dans les deux directions).

Duaux [4 points]

Donnez pour chaque q, le plus grand ϵ_q que vous puissiez pour lequel il existe une famille de codes $\{C_n\}_{n>n_0}$ et sa famille de codes duaux $\{C_n^\perp\}_{n>n_0}$ constituées de codes linéaires $C_n, C_n^\perp \subseteq \mathcal{F}_q^n$ qui soient tous deux de distance minimale $\epsilon_q n$. Justifiez votre réponse. Quelle est la valeur de $\lim_{q\to\infty} \epsilon_q$?

Donnez le plus petit ϵ que vous puissiez pour lequel il n'existe pas de famille de codes $\{C_n\}_{n>n_0}$ et sa famille de codes duaux $\{C_n^\perp\}_{n>n_0}$ constituées de codes linéaires $C_n, C_n^\perp \subseteq \mathcal{F}_2^n$ dont la distance minimale **moyenne** des deux soit ϵn. Justifiez votre réponse.

Codes Intersectants [8 points]

On dit qu'un code C engendré par $\mathbf{G}$ est *intersectant* si pour toute paire de mots de codes $c_0, c_1 \in C - \{\mathbf{0}\}$ il existe une position i où $c_0^i \neq 0 \neq c_1^i$. Soit $\mathbf{G}$ la matrice génératrice $n \times k$ d'un code *intersectant* sur $\mathcal{F}_2$.Par exemple la matrice $\mathbf{G} = \begin{bmatrix} 1\,1\,0 \\ 0\,1\,1 \end{bmatrix}$ engendre un code intersectant puisque ses mots de codes sont $000, 110, 011, 101$ et que $110, 101$ intersectent en position 1, $110, 011$ intersectent en position 2, et $011, 101$ intersectent en position 3.

1. Montrez que deux mots binaires c_0, c_1 intersectent si et seulement si $w(c_0 \oplus c_1) \neq w(c_0) + w(c_1)$.
2. Donnez un code intersectant binaire de longueur $n = 18$ et dimension $k = 6$.
3. Montrez que pour un $[n, k, d]$ code intersectant binaire on a toujours $d \geq k$.
4. Montrez qu'asymptotiquement il n'existe pas de code intersectant binaire pour $n < 3.53k$.
5. Montrez qu'asymptotiquement il existe un code intersectant binaire pour $n > 4.81k$.

IFT6170 Devoir 4

à rendre au début du cours le mercredi 3 décembre 1997

Partage de secret [6 points]

Soit un code Reed-Solomon sur $\mathcal{F}_q$ ayant des paramètres $[N = q-1, K, N-K+1]$. On suppose que ses mots de code sont définis de façon standard

$$c = (c_0\ c_1\ \dots\ c_{N-1}) = \big(F(1), F(\alpha), ..., F(\alpha^{N-1})\big)$$

pour les polynômes F sur $\mathcal{F}_q$ de degré $< K$. Supposons qu'Alice dispose d'un secret $s \in \mathcal{F}_q$. Elle peut choisir un polynôme aléatoire

$$F(x) = s + f_1 x + f_2 x^2 + \cdots + f_{K-1} x^{K-1}$$

tel que $F(0) = s$, où chaque f_i est sélectionné au hasard dans $\mathcal{F}_q$. Soit $P_0, P_1, ..., P_{N-1}$ des personnes distinctes à qui Alice va distribuer des "ombres" de son secret: Alice donne c_i à P_i. Soit $\Phi \subseteq \{0, 1, ..., N-1\}$.

- Montrez que si $|\Phi| < K$ alors $\mathbf{I}(s; \{(i, c_i) : i \in \Phi\}) = 0$. C'est à dire que tout ensemble de personnes de taille inférieure à K n'a aucune information sur s.
- Montrez que si $|\Phi| \geq K$ alors $\mathbf{H}(s \mid \{(i, c_i) : i \in \Phi\}) = 0$. C'est à dire que tout ensemble de personnes de taille au moins K peut déterminer s.
- Supposons que toutes les personnes se rencontrent et échangent leurs c_i. Considérons qu'un certain nombre t d'entre eux soit menteur et révèlent au contraire $\tilde{c}_i \neq c_i$. Donnez une relation entre N, K et t qui donne les situations où l'on sera capable d'identifier les menteurs et de récupérer quand même le secret s. Comment peut-on identifier les menteurs efficacement?

Justesen généralisés [6 points]

Considérez sur $\mathcal{F}_2$ les codes $C_{\alpha,\beta} = \{[u, \alpha u, \beta u] : u \in \mathcal{F}_{2^m}\}$ pour $\alpha, \beta \in \mathcal{F}_{2^m} - \mathbf{0}$.

- Montrez qu'il existe parmis les codes $C_{\alpha,\beta}$ des codes de paramètres $[3m, m, 3\gamma m]$ pour $\gamma \geq h^{-1}\left(\frac{2}{3}\right) \approx 0.1739523$.
- Montrez qu'une construction semblable à celle de Justesen nous permet de construire à partir des codes $C_{\alpha,\beta}$ des codes sur $\mathcal{F}_2$ ayant des paramètres $[N, RN, \delta N]$ pour $N = 3m2^m$, $0 \leq R \leq \frac{1}{3}$ et $\delta \geq (1 - 3R)h^{-1}\left(\frac{2}{3}\right)$.
- Montrez qu'une construction semblable à la précédente nous permet de construire pour chaque entier a des codes sur $\mathcal{F}_2$ ayant des paramètres $[N, RN, \delta N]$ pour $N = am2^m$, $0 \leq R \leq \frac{1}{a}$ et $\delta \geq (1 - aR)h^{-1}\left(\frac{a-1}{a}\right)$.
- Tracez ces droites sur un graphique et montrez qu'elles forment un ensemble de tangentes à la courbe de Zyablov.

Codes réversibles [4 points]

Un code $C \subseteq \mathcal{F}_q$ est *réversible* si $c_0 c_1 \dots c_{n-1} \in C$ implique $c_{n-1} c_{n-2} \dots c_0 \in C$ également. Montrez qu'un code cyclique $C = \langle g \rangle$ est réversible si et seulement si $g(\alpha) = 0$ implique toujours $g(\alpha^{-1}) = 0$. Montrez qu'un code cyclique C est réversible si et seulement si -1 est une puissance de q modulo n.

IFT6170 Devoir 5

à rendre avant la tombée de la nuit le vendredi 19 décembre 1997

Codes BCH sont Alternants [8 points]

Montrez de façon générale que les codes BCH sont des codes alternants. C'est-à-dire que les mots de codes peuvent être obtenus par la restriction sur $\mathcal{F}_q$ des mots d'un code RSG(α, v) sur $\mathcal{F}_{q^m}$.

Prenez le code BCH binaire de longueur 23 et de distance de design 3. Un polynôme générateur de ce code est $M^{(1)}(x)$. La dimension de ce code est de 12 et sa distance minimale réelle est de 7. En utilisant votre construction générale donnez le code RSG(α, v) dont ce code BCH est la restriction sur $\mathcal{F}_2$.

Calculez le syndrome S du patron d'erreur (01000000000000000000000). Trouvez les polynômes σ et ω associés à ce patron. Vérifiez que $\omega(x) \equiv \sigma(x)S(x) \pmod{x^r}$.

Codes Intersectants [8 points]

1. Si on étend la définition de codes intersectants à tous les corps $\mathcal{F}_q$ il est possible sur certains corps de construire explicitement de tels codes. Montrez que pour des $q > 2$ assez grand certains codes Reed-Solomon sont intersectants sur $\mathcal{F}_q$.
   suggestion: Tout code $[n, k, d]$ pour d assez grand est intersectant.
2. Pourquoi est-ce que la **suggestion** du point **??** ne nous aide pas à montrer directement qu'il existe une bonne famille de codes intersectants sur $\mathcal{F}_2$.
3. Montrez que la concaténation de deux codes intersectants est également un code intersectant.
4. Expliquez comment combiner les points précédents et ce que l'on connais déja sur les codes intersectants pour obtenir une construction efficace probabiliste d'une bonne famille de codes binaires (vérifiablement) intersectants. Donnez de façon explicite le lien entre k et n des codes binaires ainsi obtenus.

References

1. ALEXI, W., B. CHOR, O. GOLDREICH, AND C. P. SCHNORR, "RSA and Rabin Functions: Certain Parts Are as Hard as the Whole". In *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, 1984, pp. 449-457.
2. ANDERSON, R. J., "Why Cryptosystems Fail", In *Proceedings of the 1993 ACM Conference in Computer and Communications Security*, pp. 215–227.
3. BENNETT, C.H., G. BRASSARD, C. CRÉPEAU AND M.–H. SKUBISZEWSKA, "Practical Quantum Oblivious Transfer", In Advances in Cryptology: Proceedings of Crypto '91, Lecture Notes in Computer Science, August 1991, Springer-Verlag, Berlin, pp. 351–366.
4. BLUM, M., L. BLUM AND SHUB, "A simple unpredictable pseudo-random number generator", *SIAM J. Computing*, 15(2):364–383, May 1986.
5. BRASSARD, G., "On computationally secure authentication tags requiring short secret shared keys", Advances in Cryptology: Proceedings of CRYPTO 82, Plenum Press, 1983, pp.79–86.
6. BRASSARD, G., C. CRÉPEAU, AND J.–M. ROBERT, "Information theoretic reductions among disclosure problems", In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, November 1986, pp. 168 – 173.
7. CARTER, J.L., M. N. WEGMAN, "New Hash Functions and Their Use in Authentication and Set Equality", Journal of Computer and System Sciences, Vol. 22, 1981, pp. 265–279.
8. CRÉPEAU, C., "Equivalence between two flavours of oblivious transfers", In Advances in Cryptology: Proceedings of Crypto '87, Lecture Notes in Computer Science, August 1987, pp. 350 – 354.
9. CRÉPEAU, C., "Quantum Oblivious Transfer", Journal of Modern Optics, Dec. 1994.
10. CRÉPEAU, C. AND J. KILIAN, "Achieving oblivious transfer using weakened security assumptions", In *29^{th} Symposium on Foundations of Computer Science*, pages 42–52. IEEE, 1988.
11. CRÉPEAU, C. AND L. SALVAIL, "Quantum Oblivious Mutual Identification", Advances in Cryptology: Proceedings of EUROCRYPT '95,Springer-Verlag, 1995.
12. CRÉPEAU, C. AND M. SÁNTHA, "Efficient reductions among oblivious transfer protocols based on new self-intersecting codes", In Sequences II, Methods in Communications, Security, and Computer Science, pp. 360–368. Springer-Verlag, 1991.
13. EVEN, S., O. GOLDREICH, AND A. LEMPEL, "A randomized protocol for signing contracts", *Communications of the ACM*, vol. 28, 1985, pp.637–647.
14. FAGIN, R., M. NAOR AND P. WINKLER, "Comparing Common Secret Information without Leaking it", submitted for publication, Communications of the ACM, 1994.
15. FIAT, A. AND A. SHAMIR, "How to prove yourself: practical solutions to identification and signature problems", In Advances in Cryptology: Proceedings of Crypto '86, Lecture Notes in Computer Science, August 1986 pp. 186–194. Springer, 1987.
16. GOLDREICH, O., S. GOLDWASSER, AND S. MICALI, "How to construct random functions", *Journal of the ACM*, 133:792–807, 1986.
17. GOLDREICH, O., S. MICALI AND A. WIGDERSON, "How to play any mental game, or: A completeness theorem for protocols with honest majority", In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, May 1987, pp. 218 – 229.
18. HÅSTAD, J., "Pseudo-random generation under uniform assumptions", *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, May 1990, pp. 395 – 440.
19. IMPAGLIAZZO, R., L. A. LEVIN AND M. LUBY, "Pseudo-random generation from one-way functions", *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, May 1989, pp. 12 – 24.
20. KILIAN, J., "Founding cryptography on oblivious transfer", In Proc. 20th ACM Symposium on Theory of Computing, pp. 20–31, Chicago, 1988. ACM.
21. MAC WILLIAMS, F. J. AND N. J. A. SLOANE, "The Theory of Error-Correcting Codes", North-Holland, 1977.
22. RABIN, M. O., "How to exchange secrets by oblivious transfer", Technical Memo TR–81, Aiken Computation Laboratory, Harvard University, 1981.
23. SPIELMAN, D., "Linear-Time Encodable and Decodable Error-Correcting Codes", In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 1995.
24. "One Less Thing to Believe In: Fraud at Fake Cash Machine", New York Times, 13 May 1993, pp. A1 & B9.
25. WIESNER, S., "Conjugate coding", *Sigact News*, vol. 15, no. 1, 1983, pp. 78 – 88; Manuscript written *circa* 1970, unpublished until it appeared in SIGACT News.
26. YAO, A. C.–C., "Protocols for secure computations", In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, November 1982, pp. 160 – 164.

This article was processed using the LaTeX macro package with LLNCS style