Résumé : Sciences de l'Information

Olivier Cloux

Printemps 2015

Contents

0 Introduction

Notes de sécurité

Le présent document n'est pas un résumé exhaustif ni officiel du cours, il ne donne pas de connaissances aussi précises que celles obtenues en suivant le cours. Je vous conseille très fortement d'utiliser le présent document comme un pense-bête, un aide mémoire, mais de s'appliquer à comprendre la matière avec le polycopié de référence, les slides et les séries d'exercices. Très peu de preuves sont présentées ici, et peuvent être testées dans un examen.

Le document ne se veut pas non plus d'une exactitude irréprochable. Des fautes, coquilles et petites erreurs sont certainement présentes. Merci de m'en avertir d'une quelconque manière, afin que je puisse corriger le document et vous fournir une version mise à jour.

À titre personnel, je vous rappelle de :

- Lire et comprendre le polycopié de référence.
- Lire et comprendre le cours (slides).
- Lire et comprendre la plupart des preuves fournies.
- Faire les séries d'exercice fournies, aussi de l'année précédente et les examens blancs fournis.
- Si vous avez des questions, bien que je ne sois pas un assistant je crois comprendre la matière, donc c'est avec plaisir que je vous file un coup de main :)

Note quant à l'impression : Le présent document comporte des hyper-références (en violet), c'est à dire des liens sur des mots ou des chiffres qui rapportent à une autre partie du document. Cliquer sur ces mots en violet vous amènera à la partie concernée. En imprimant vous perdez la possibilité de vous servir de ces références, mais le document reste tout à fait compréhensible

Notes quant à la rédaction Ce document a été rédigé en LaTeXsur l'éditeur TexMaker pour Linux. Je l'ai écrit sur tout un semestre, j'ai donc changé mon style de rédaction, et j'ai appris plein de choses en LaTeXdepuis le début. Malgré que je me sois efforcé de rendre le document uniforme, des inhomogénéités peuvent frapper l'œil averti. Toutes les critiques sont bienvenues ! Et bien évidemment, bien que je ne sois pas un gourou du LaTeX, je me débrouille assez pour donner quelques conseils. Si vous voulez une partie/l'intégralité de mon code source, n'hésitez pas à demander, on en discutera.

Maintenant que tout cela est dit, j'espère que mon document vous plaira, et que vous le trouverez assez clair et précis. N'hésitez pas à m'envoyer un mot de remerciements si vous avez apprécié, ça fait toujours plaisir et ça me motive à garder le document à jour :)

D'avance merci pour votre lecture, et bonne chance pour vos examens !!

Olivier Cloux

Note pour tout le document: sauf mention contraire, tout ce qui s'applique à 2 marginales/sources/... s'applique aussi à n

1 Chapitre 1 : Information, probabilité, entropie

Exemple 1. Nous allons utiliser deux types de sources pour tous nos exemples : les tirages de Bart et de Lisa selon le modèle suivant :

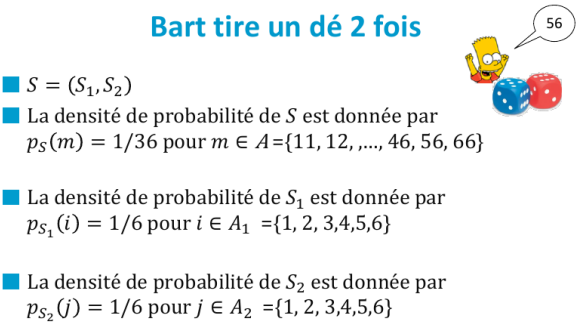
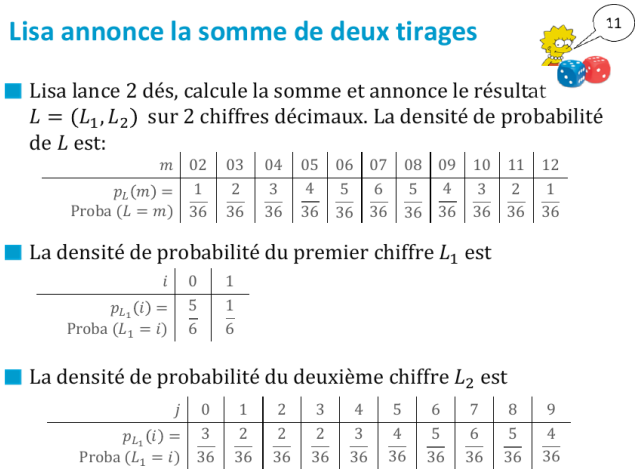


Figure 1: Les tirages de Bart



m	02	03	04	05	06	07	08	09	10	11	12
$p_L(m)$ = Proba $(L = m)$	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

i	0	1
$p_{L_1}(i)$ = Proba $(L_1 = i)$	$\frac{5}{6}$	$\frac{1}{6}$

j	0	1	2	3	4	5	6	7	8	9
$p_{L_1}(i)$ = Proba $(L_1 = i)$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$

Figure 2: Les tirages de Lisa

1.1 Probabilités

La source S produit un message $m \in A$ (= Alphabet). La probabilité du message m est $0 \leq p(m) \leq 1$ est définie par $\frac{\text{cas favorables}}{\text{cas possibles}}$. La fonction $m \to p(m)$ est appelée la densité de probabilité.

Une source composée est

$$\overbrace{S}^{\text{source composée}} = \underbrace{(S_1, S_2, ...)}_{\text{Sources marginales}}$$

Exemple 2. Bart tire un dé deux fois. Ainsi $S = (S_1, S_2)$. L'ensemble constitué des deux tirages est une source composée des deux sources marginales, chacun des tirs.
S peut prendre 36 formes : (11,12,13,...,16,21,22,...,66), toutes équiprobables. Ainsi, $p_S(m) = \frac{1}{36}$ pour $m \in \{11, 12, ..., 65, 66\}$ et $p_{S_1}(i) = \frac{1}{6} = p_{S_2}(j)$ pour $i, j \in A_1, A_2$ et $A_1 = A_2 = \{1, 2, 3, 4, 5, 6\}$

En revanche, Lisa tire deux dés et annonce la somme des tirs. Les densités de probabilités ne sont alors plus équiprobables (probabilité de 7 beaucoup plus grande que celle de 12). Le tableau des probabilités est donné à la Figure 2. De plus, le premier chiffre ne peut être que 0 ou 1 (car les résultats vont de 02 a 12), alors que le second chiffre va de 1 à 9, d'où une certaine inhomogénéité dans les probabilités.

1.1.1 Indépendance

Deux marginales sont indépendantes ssi, pour tout i, j,

$$p(S_1 = i,\ S_2 = j) = p(S_1 = i)p(S_2 = j)$$

Exemple 3. La densité de probabilité de Bart est de $p_S(i,j) = \frac{1}{36}$, alors que la probabilité de ses marginales est de $\frac{1}{6}$. Ainsi, $p_S(i,j) = \frac{1}{36} = \frac{1}{6} \times \frac{1}{6} = p_{S_1}(i) \cdot p_{S_2}(j)$ donc les deux marginales sont indépendantes.
En revanche, pour Lisa, $p_L(1,2) = \frac{1}{36} \neq \frac{0.\overline{3}}{36} = \frac{1}{6} \times \frac{2}{36} = p_{L_1}(1) \cdot p_{L_2}(2)$. Donc les deux marginales ne sont pas indépendantes.

1.2 Entropie

L'entropie se calcule indépendamment du système de codage (binaire, tertiaire,...). On ne compte que les probabilités d'apparition, pour savoir combien de bits on a besoin au minimum pour représenter nos mots de source. L'entropie "encode" tout en binaire, pour mieux les comparer. Changer la base du log servirait à "encoder" en une autre base. Le calcul de l'entropie est :

$$H(S) = -\sum_{s \in A} p(s) \log_2 \big(p(s)\big) = \sum_{s \in A} p(s) \log_2 \left(\frac{1}{p(s)}\right) \tag{1}$$

A savoir : L'entropie est nulle $\iff$ la source est déterministe, c'est à dire "Il existe un symbole s tel que $p(s) = 1 \iff H(S) = 0$ et pour tous les autres symboles $s' \neq s, p(s') = 0$"
A savoir : L'entropie est maximale $\iff$ la source est uniforme ($= M \cdot \frac{1}{M} \log_2(M) = \log_2(M)$)
Théorème 1.3:

- $H(S) \leq \log_2(M)$
- Les M symboles de la source sont équiprobables $\iff$ $H(S) = \log_2(M)$

1.2.1 Indépendance

Par définition de l'entropie, pour toutes sources composées

$H(S_1, S_2) \leq H(S_1) + H(S_2)$
$H(S_1, S_2) = H(S_1) + H(S_2)$ Si les sources sont indépendantes (**Théorème 1.4**)

Exemple 4.
$$\left.\begin{array}{r} H(S) = 36 \cdot \frac{1}{36} \log_2(36) = \log_2(36) \simeq 5.170 \text{ bits} \\ H(S_1) = H(S_2) = 6 \cdot \frac{1}{6} \log_2(6) = \log_2(6) \simeq 2.585 \text{ bits} \end{array}\right\} H(S) = H(S_1) + H(S_2)$$
$\rightarrow S_1$ et S_2 sont indépendants
Mais
$$\left.\begin{array}{r} H(L) = \frac{1}{36} \log_2(36) + \frac{2}{36} \log_2(\frac{36}{2}) + ... = 3.27 \text{ bits} \\ H(L_1) = \frac{5}{6} \log_2(\frac{6}{5}) + \frac{1}{6} \log_2(6) = 0.65 \text{ bits} \\ H(L_2) = \frac{3}{36} \log_2(\frac{36}{3}) + \frac{2}{36} \log_2(\frac{36}{2}) + ... = 3.22 \text{ bits} \end{array}\right\} H(L) < H(L_1) + H(L_2) =$$
3.87 bits $\rightarrow L_1$ et L_2 sont dépendants

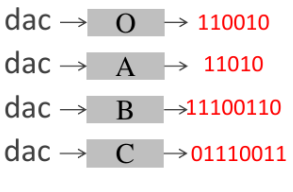
2 Chapitre 2 : Codage de Sources

2.1 Terminologie

Le code suivant sert à illustrer la terminologie et à fournir des codes d'exemples.

code	O	A	B	C
a	00	0	0	0
b	01	01	10	01
c	10	10	110	011
d	11	11	1110	0111

TABLE 2.1: Quatre codes binaires. L'alphabet de la source est $\{a, b, c, d\}$



Code A:
Alphabet de la source: $\{a, b, c, d\}$
Alphabet du code: $\{0,1\}$
Mots de code: $\{0, 01, 10, 11\}$

Figure 3: Terminologie

2.2 Décodage unique, instantané, arbres

2.2.1 Décodage unique

Un code est à décodage unique si aucune suite de symbole de code ne peut être décodée de 2 manières différentes.

Exemple 5. Par exemple, pour le code A, $bc = 0110 = ada$.

On peut identifier un code comme étant à décodage unique grâce au flair et à quelques outils :

- Tous les mots de code sont de même longueur (c.f. code O)
- Un caractère unique marque le début ou la fin de chaque mot (c.f. codes B et C, donc le 0 se trouve uniquement et respectivement à la fin et au début de chaque mot)
- S'il est à décodage instantané il est forcément à décodage unique

2.2.2 Décodage instantané

Un code est à décodage instantané si aucun mot de code n'est le préfixe d'un autre (donc si aucun mot n'est dans plus haut dans une même branche de l'arbre de décodage). Simple à vérifier, cela est très pratique. S'il est instantané, alors il est à décodage unique (attention, la réciproque est fausse).

Exemple 6. Les codes O et B sont sans préfixe, alors que les codes A et C ne le sont pas (code A : a est préfixe de b et code C : a est préfixe de tout).

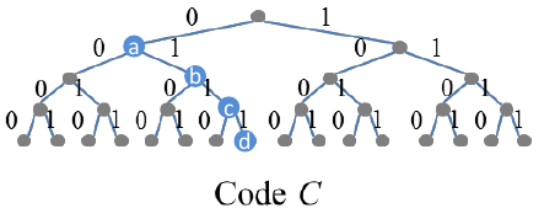


Figure 4: L'arbre complet de C

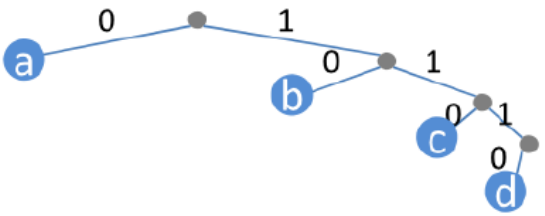


Figure 5: L'arbre de décodage de B

2.2.3 Arbres

On construit un arbre en faisant descendre des branches d'un point, chaque branche étant un symbole (0,1 pour un code binaire, 0,1,2 ternaire, etc.) Sa profondeur est la longueur des mots de code (4 dans nos 2 exemples), et à la profondeur n il y aura, pour un code D-aire, D^n mots dans l'arbre complet.
Un **arbre complet** (Fig. 4) se fait en créant tous les points et toutes les branches
Un **arbre de décodage** (Fig. 5) se fait en enlevant les branches en dessous d'un mot de code (uniquement pour code sans préfixe)

2.3 Kraft-McMillan

2.3.1 Partie 1

On parle d'un code Γ D-aire dont les longueurs des M mots de codes sont $l_1, \ldots, l_M$

$$\text{Décodage unique} \rightarrow \frac{1}{D^{l_1}} + \frac{1}{D^{l_2}} + \ldots + \frac{1}{D^{l_M}} \leq 1 \tag{2}$$

Donc si le code est binaire, D = 2.
Par contraposée, si l'inégalité n'est pas respectée, alors le code n'est pas à décodage unique. La réciproque est fausse (mais voir la partie 2).

Exemple 7. L'inégalité de Kraft n'est pas respectée pour A car

$$2^{-1} + 2^{-2} + 2^{-2} + 2^{-2} = 1.25 > 1$$

Donc le code n'est pas à décodage unique. Mais elle l'est pour C car

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 0.9375 \leq 1$$

On ne peut cependant pas conclure que C est à décodage unique (démontré avant).

2.3.2 Partie 2

Même si la réciproque directe est fausse, la seconde partie dit que si des nombres $l_1, \ldots l_M$ satisfont l'inégalité de Kraft, alors un code à décodage instantané est possible avec ces longueurs.

$$\frac{1}{D^{l_1}} + \frac{1}{D^{l_2}} + \ldots + \frac{1}{D^{l_M}} \leq 1 \rightarrow \begin{array}{c}\text{Il existe un code D-aire instantané donc le dictionnaire posède} \\ \text{M mots de code et dont les longueurs des mots de code sont } l_1,\ldots,l_M\end{array}$$

Exemple 8. Même si A n'est pas à décodage unique, en considérant un code A' similaire à A mais ternaire, l'inégalité est respectée, car

$$3^{-1} + 3^{-2} + 3^{-2} + 3^{-2} = 2/3 \leq 1$$

Mais le code n'est toujours pas à décodage unique (toujours parce que $bc = ada$). Nous savons cependant qu'il existe un code avec ces longueurs qui est à décodage unique. Ainsi, en changeant $b = 10$ en 12, le code est à décodage unique (et même instantané).

2.3.3 Construction d'un arbre de Kraft

- On construit un arbre D-aire de profondeur égal à $l_{\max}$
- Placer les mots par ordre croissant
- Supprimer les branches en dessous

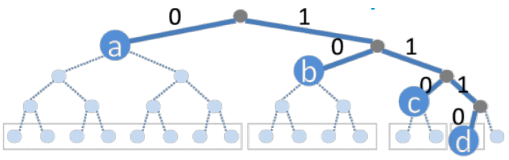
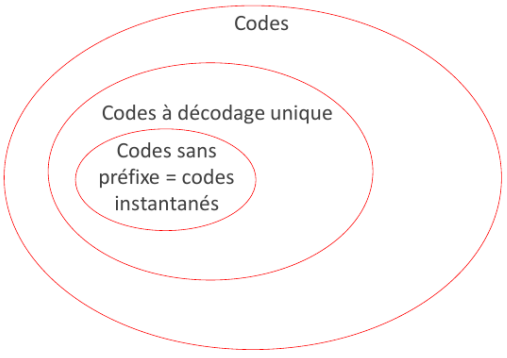


Figure 6: Construction de l'arbre de Kraft pour le code B

2.4 Définitions, Théorèmes et le reste

La **longueur** se définit par le nombre de symboles de sources dans un mot de code.
L'ensemble des codes respecte ce graphique :

Figure 7: Ensemble des codes



De manière similaire à la seconde partie de l'inégalité de Kraft, le **Théorème 2.3** dit que pour tout code à décodage unique, il existe un code à décodage instantané sur les mêmes alphabets de source et de code qui a les mêmes longueurs de mots.

Exemple 9. C'est ce qu'on fait en remplaçant le code C par le code B

3 Chapitre 3 : Efficacité d'un code de source

3.1 Longueur moyenne

Pour une Source S, de densité de probabilité p, et Γ un code D-aire. Alors la longueur moyenne est :

$$L(\Gamma) = \sum_{s \in A} p(s) l(\Gamma(s)) \tag{3}$$

Avec pour unité le "symbole de code par symbole de source" (Si D = 2 on parle de *bits* par symbole de source)

Exemple 10.

source	a	b	c	d
mots de code	0	10	110	1110
p(s)	0.1	0.4	0.3	0.2

Avec cette distribution de probabilités, la longueur moyenne est alors
$L(\Gamma) = 0.1 \cdot 1 + 0.4 \cdot 2 + 0.3 \cdot 3 + 0.2 \cdot 4 = 2.6$

3.2 Première inégalité de l'entropie

Pour un code D-aire :

$$\text{Décodage unique} \rightarrow L(\Gamma) \geq \frac{H(S)}{\log_2(D)} \tag{4}$$

3.2.1 Remarques

Pour un code binaire : $\log_2(D) = 1$, donc l'inégalité deviens $L(\Gamma) \geq H(S)$. Cela se traduit par la remarque suivante : On ne peut pas faire mieux que l'entropie

3.3 Seconde inégalité de l'entropie

Un code D-aire de Shannon-Fano doit vérifier

$$\frac{H(S)}{\log_2(D)} \leq L(\Gamma_{SF}) < \frac{H(S)}{\log_2(D)} + 1 \tag{5}$$

3.3.1 Remarque

Bien sur, si le code est binaire alors $\log_2(D) = 1$ donc on peut ôter les log de notre équation ; on comprend alors "Shannon-Fano est entre 'entropie et entropie + 1'"

3.4 Code/arbre de Shannon-Fano

Données les probabilités d'apparition de chaque mot du dictionnaire, il devient facile de créer un bon encodage :

1. On calcule les longueurs de chaque mot avec $\lceil \log_2 \left(\frac{1}{p_i} \right) \rceil$
2. Ça nous donne des longueurs arrondies pour chaque mot.
3. Ensuite on place dans un arbre, aux bonnes positions

Exemple 11. Prenons un code B' aux probabilités suivantes :

symbole de source	proba
a	0.05
b	0.05
c	0.1
d	0.8

Il faut donc que les longueurs des mots de code de Shannon soient les suivantes :
$a, b : \lceil -\log_2(0.05) \rceil = \lceil 4.3219 \rceil = 5$
$c : \lceil -\log_2(0.1) \rceil = \lceil 3.3219 \rceil = 4$
$d : \lceil -\log_2(0.8) \rceil = \lceil 0.3219 \rceil = 1$
Et depuis là on place dans un arbre aux positions voulues, ce qui donnerait (par exemple) $a = 00001,\ b = 00000,\ c = 0001,\ d = 1$

3.4.1 Remarques

Le code est toujours "assez bon" mais rarement optimal ; il l'est quand il n'y a pas d'arrondi, donc tous les $p(s)$ sont des puissances de 2 (comme $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, ...$). Ce code est utilisé dans la seconde inégalité de l'entropie. Le code est instantané.

Exemple 12. Avec la méthode de Huffman, le code de l'exemple précédent aurait eu des longueurs de respectivement 3,3,2,1, ce qui est plus efficace.

3.5 Code/arbre de Huffman

1. Placer les mot de code côte à côte triés par probabilités d'apparition
2. Relier les deux plus faibles probabilités
3. Additionner leurs probabilités
4. Recommencer jusqu'à ce que tous soient réunis

Exemple 13. Reprenons les probabilités de notre code B'. La construction se l'arbre de Huffman se trouve à la Figure 8. Nous n'avons alors qu'à lire la position de chaque mot pour trouver son encodage, qui est

a	111
b	110
c	10
d	0

3.5.1 Remarques

- Le code est optimal : $L(\Gamma_H) \leq L(\Gamma)$ pour tout autre code binaire à décodage unique
- Le code est instantané
- Construit un code binaire ! La méthode ne fonctionne a priori pas pour construire un code ternaire ou plus.

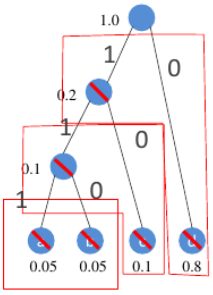


Figure 8: L'arbre de Huffman du code B'

4 Chapitre 4 : Entropie conditionnelle

Exemple 14. Comme les sources de Bart sont indépendantes, on ne peut pas (pas besoin de) travailler avec l'entropie conditionnelle. En effet, la probabilité de $S_2 = j$ sachant que $S_1 = i$ vaut la probabilité de $S_2 = j$. Par contre le cas de Lisa est plus intéressant, donc c'est sa manière de faire qu'on utilisera dans les exemples, comme à la Figure 9

	i	0	1	
j	$p_L(i,j)$			$p_{L_2}(j)$
0		0	3/36	3/36
1		0	2/36	2/36
2		1/36	1/36	2/36
3		2/36	0	2/36
4		3/36	0	3/36
5		4/36	0	4/36
6		5/36	0	5/36
7		6/36	0	6/36
8		5/36	0	5/36
9		4/36	0	4/36
	$p_{L_1}(i)$	5/6	1/6	

Figure 9: Le tableau des probabilités conditionnelles $(L_2|L_1)$ de Lisa

Ce tableau n'est pas simple à lire : la colonne du milieu représente le premier chiffre (qu'on rappelle, est la somme du tirage de deux dés). Les lignes représentent le second chiffre. On a donc aucune chance d'avoir un 0 annoncé en second chiffre si le premier est déjà 0 (elle ne peut pas annoncer 00). Par contre le chiffre 02 a une probabilité 1/36 d'arriver (somme de 1 et 1), et 07 a 6/36 chances d'arriver.

4.1 Probabilité conditionnelle

"Probabilité que la source S_2 émette le symbole s_2 sachant que la source S_1 émet le symbole s_1" = $p_{S_2|S_1}(s_2|s_1) = \frac{p(s_1,s_2)}{p(s_1)}$

Exemple 15. Probabilité que le second chiffre annoncé soit 2 sachant que le premier est 1 :

$$p_{L_2|L_1}(2|1) = \frac{p_{L_1,L_2}(12)}{p_{L_1}(1)} = \frac{\frac{1}{36}}{\frac{1}{6}} = \frac{1}{6}$$

Et on voit que cela correspond à la 3ème ligne à la colonne "1" du tableau. En faisant les mêmes calculs pour une autre case, ça fonctionne toujours (par exemple $p_{L_2|L_1}(2|0) = \frac{1}{30}$)

4.2 Entropie conditionnelle

L'entropie conditionnelle de S_2 sachant $S_1 = s_1$ est l'entropie de la densité conditionnelle de S_2 sachant $S_1 = s_1$:

$$H(S_2|S_1 = s_1) = -\sum_{s_2 \in A_2} p_{S_2|S_1}(s_2|s_1) \log_2 \left(p_{S_2|S_1}(s_2|s_1)\right) \tag{6}$$

Exemple 16.

$$H(L_2|L_1 = 1) = \frac{3}{6}\log_2\left(\frac{6}{3}\right) + \frac{2}{6}\log_2\left(\frac{6}{2}\right) + \frac{1}{6}\log_2\left(\frac{6}{1}\right) \approx 1.459 \text{ bits}$$

$$H(L_2|L_1 = 0) = \frac{1}{30}\log_2\left(\frac{30}{1}\right) + \frac{2}{30}\log_2\left(\frac{30}{2}\right) + \ldots + \frac{4}{30}\log_2\left(\frac{30}{4}\right) \approx 2.857 \text{ bits}$$

Et l'entropie (générale, pas dans un cas) conditionnelle de S_2 sachant S_1 en est la moyenne :

$$H(S_2|S_1) = \sum_{s_1 \in A_1} H(S_2|S_1 = s_1) p_{s_1}(s_1) \tag{7}$$

Exemple 17. Rappelons que la probabilité que le premier chiffre soit 0 est de 5/6 et qu'il soit 1 est de 1/6. Ainsi :

$$H(L_2|L_1) = \frac{1}{6}H(L_2|L_1 = 1) + \frac{5}{6}H(L_2|L_1 = 0) = \frac{1}{6}1.459 + \frac{5}{6}2.587 = 2.624 \text{ bits}$$

Nous avons vu à l'exemple 4 que $H(L_1) = 0.65$. Or, $H(L_1) + H(L_2|L_1) = H(L_1, L_2)$ (c.f. la règle de l'enchaînement)

Cela s'interprète comme étant la quantité d'information moyenne supplémentaire, la quantité d'information reçue en observant S_2 après avoir observé S_1.

Exemple 18. Le point à bien saisir ici, c'est que l'équation (6) regarde dans un cas précis ; pour reprendre les tirages de Lisa, cette équation doit se faire beaucoup de fois : par exemple, quelle est l'entropie du second chiffre *sachant que le premier est 0.* L'équation (7) regarde la moyenne de tous les cas : quelle est l'entropie du second chiffre de Lisa *en admettant que je connaisse le premier*

4.2.1 Règle de l'enchaînement

Pour une source composée :

$$\begin{aligned} H(S_1, S_2) &= H(S_1) + H(S_2|S_1) \\ &= H(S_2) + H(S_1|S_2) \end{aligned}$$

(Donc information totale = information d'un + l'information supplémentaire)
Peut aussi dans l'autre sens ($H(S_2|S_1) = H(S_1, S_2) - H(S_1)$)

4.2.2 Conditionner réduit l'entropie

Pour une source composée :
$H(S_2|S_1) \leq H(S_2)$
$H(S_2|S_1) = H(S_2)$ si et seulement si S_1 et S_2 sont indépendants

4.3 Remarques, définitions, théorèmes

Une source est fonction d'une autre si connaître le premier implique le second. Par exemple, tirer deux dés (S_1, S_2) et calculer leur somme K. K est alors fonction de (S_1, S_2), car K, n'apporte aucune information supplémentaire par rapport à (S_1, S_2) $\Rightarrow$ S_2 est *fonction* de S_1 si $H(S_2|S_1) = 0$

5 Chapitre 5 : Théorème du Codage de Sources

Note personnelle Ce chapitre est vraiment très compliqué (à mon goût le plus difficile), les définitions sont floues et la série avec était horrible ; il est donc difficile de retranscrire toute la matière sans la recopier mot-à-mot. Une partie essentielle de la compréhension est passée par les question avec clickers. Je vous conseille pour ce chapitre de vraiment relire les slides et le livre, mon résumé ne transmettant que peu ce qui est transmis dans le cours. Refaire/lire la série 4 et son corrigé est une bonne idée aussi.

5.1 Définitions, théorèmes

Bloc : Un bloc produit une suite de n symboles venant chacun de l'alphabet A. L'alphabet de S^n est A^n

Source étendue : Une source étendue est un bloc infini. Pour le vérifier, il faut vérifier 2 points précis :

- S^n est une source à n composantes, sur l'alphabet $\underbrace{A \times A \times \ldots \times A}_{\times n}$; notons p_{S^n} sa densité de probabilité.
- La densité de probabilité de la source constituée des n premières sources marginales de S^{n+k} est égale à p_{S^n}, $\forall k, n \geq 1$

Si ces deux items sont respectés, la source est étendue.

Source stationnaire Pour qu'une source soit stationnaire, il faut vérifier si la densité de probabilité $p_{S_{k+1},S_{k+2},\ldots,S_{k+n}}(s_1, s_2, ..., s_n)$ est la même pour toutes les valeurs de $k \geq 0$. Autrement dit, le fait de regarder la densité de probabilité d'un bloc de longueur n (fixe par cas), ne devrait pas changer, quel que soit l'endroit où on regarde (par exemple regarder tous les bloc de taille 20 ; regarder au $3^{\text{ème}}$ ou $1000^{\text{ème}}$ tirage ne devrait rien changer). Pour cela, il faut calculer la probabilité (de longueur n) du bloc ; si la probabilité dépend d'un k, alors la source n'est pas stationnaire.

Entropie d'un symbole : $H(S) = \lim_{n \to +\infty} H(S_n)$ (attention, entropie du $n^{\text{ème}}$ symbole, pas du bloc de taille n)

Entropie par symbole : $H^*(S) = \lim_{n\to+\infty} H(S_n|S_1, S_2, \ldots, S_{n-1})$

Source régulière : Une source est régulière si les limites $H(S)$ et $H^*(S)$ existent et sont finies

Bits par symboles ($\neq$ entropie par symbole) : $\frac{L_n}{n}$

Entropie d'un bloc Soit S source étendue régulière, on peut poser que $\lim_{n\to+\infty} \frac{H(S_1,S_2,\ldots,S_n)}{n} = H^*(S)$

Théorème 5.3 : (Codage de Source) Soit S une source étendue régulière et $H^*(S)$ son entropie par symbole. Soient L^n_{SF}, respectivement L^n_H, les longueurs moyennes des codes D-aires de Shannon-Fano, respectivement Huffman, pour un bloc de n symboles de la source. Alors

$$\lim_{n\to\infty} \frac{L^n_H}{n} = \lim_{n\to\infty} \frac{L^n_{SF}}{n} = \frac{H^*(S)}{\log_2(D)}$$

Autrement, pour une source régulière et un code optimal, la longueur moyenne divisée par la taille du bloc n'est autre que l'entropie par symbole.
Notons que cela implique une chose intéressante : Pour un code binaire : $\lim_{n\to\infty} \frac{L^n_H}{n} = H^*(S)$
Définitions :

- Toutes les sources stationnaires sont régulières
- **Théorème 5.1 :** Pour une source régulière, $H^*(S) \leq H(S)$ (si les sources marginales sont indépendantes, alors il y a égalité)

5.2 Rapport de compression

code B'	symbole de source s	proba $p(s)$	code Γ_H
1110	a	0.05	111
110	b	0.05	110
10	c	0.1	10
0	d	0.8	0

code	O
a	00
b	01
c	10
d	11

$L(B') = 1.35$

$L(\Gamma_H) = 1.30$

Le rapport de compression de B' est $\frac{1.35}{2} = 0.675$

Figure 10: Un exemple de code

Le rapport de compression d'un code se calcule en divisant sa longueur moyenne par la longueur moyenne d'un code "stupide", (un code de longueur constante la plus petite possible), obtenu par

$$n\lceil \log_2(M) \rceil$$

. Cela nous donne que le rapport de compression est

$$\frac{L^n}{n\lceil \log_2(M) \rceil} (\leq 1)$$

Dans l'exemple de la Figure 10, le code O est le code stupide. Sa longueur est de 2, raison pour laquelle on divise la longueur moyenne de B' par 2
Nous savons aussi, grâce aux entropies, que le rapport de compression d'un code optimal est

$$\sim \frac{H^*(S)}{\log_2(D)}$$

5.3 Régulière, stationnaire, étendue : quoi et pourquoi ?

- **Régulière** : On a vu l'importance de $H(S_n)$ et de $\frac{H(S_1,\dots,S_n)}{n}$. On aimerait savoir si, pour la source donnée, ces quantités se stabilisent pour n assez grand. C'est le cas des sources régulières.

- **Stationnaire** : Pour la plupart des sources, les propriétés statistiques ne dépendent pas d'un "décalage d'indexation".

Exemple 19. Si on observe

$$[...]101101[...]$$

est-ce une réalisation de

$$[...]S_1S_2S_3S_4S_5S_6[...]$$

ou de

$$[...]S_{n+1}S_{n+2}S_{n+3}S_{n+4}S_{n+5}S_{n+6}[...]?$$

Aucune différence statistique si stationnaire

- **Étendue** :On nous donne une *famille* de sources. Cela forme-t-il une source étendue $S_1, S_2, S_3, \dots$?

Exemple 20. Prochaine nouveauté, le display triangulaire : le iDisplay (Figure 11). Cela ne donne pas une source étendue, car

$$\sum_{i\in \text{ couleurs}} \overbrace{P_{S^2}(\text{vert}, i)}^{0} = 0$$
$$\neq P_{S^1}(\text{vert}) = 1$$

Dit autrement, S_2 n'est pas une source étendue du type (S^1, S^2) etc.

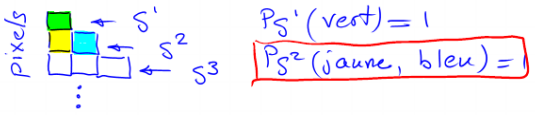


Figure 11

5.4 Ne pas confondre, pour un code binaire

$$\text{Rapport de compression } = \frac{L^n}{n\lceil \log_2 M\rceil} \neq \frac{L^n}{n} = \text{ bits pas symbole}$$

même si leur valeur est identique, car le code est binaire

6 Chapitre 6 : La cryptographie

En cryptographie nous avons essentiellement 2 types d'encodage : symétrique et asymétrique. Le codage symétrique a l'avantage d'être presque incassable (alors que l'asymétrique est sujette à plus d'attaques), mais le transfert de la clé secrète est difficile. Si on a la clé on peut tout savoir ; la difficulté devient alors de transmettre cette clé sur un canal sécurisé. Le cryptage asymétrique utilise en revanche une clé secrète et une clé publique.

6.1 Terminologie

- P : le *Plaintext*, le texte en clair, celui de départ.
- C : le *cyphertext*, le texte codé, celui que l'on transmet (qui risque d'être intercepté).
- K : la clé (*Key*) de cryptage.
- k : la clé de décryptage
- $E_k(P)$: l'*Encodage* qui utilise la clé k pour encoder P. C'est la fonction de P à C.
- Symétrique : Si $K = k$. On garde alors cette clé secrète, et il faut décrypter d'un autre moyen (par exemple brute-force, ou convenu longtemps à l'avance).
- Asymétrique : Si $K \neq$ k. À ce moment, soit on rend K publique et on garde k secrète (confidentialité) soit l'inverse (authentification)

Par principe, nous devons supposer que l'intrus intercepte tout, comme notre destinataire, qu'il l'intercepte à la transmission (le message est alors codé), et qu'il sait quelle méthode a été utilisée (se baser sur le manque de connaissance de l'intrus est extrêmement risqué et ne constitue pas une sécurité à long terme).

6.2 Chiffre de César

Cryptage symétrique. On prend un chiffre (modulo 26), et on décale les lettres P de cet ordre.

Exemple 21. Par exemple, si K = 3, alors $A \to D, B \to E, ..., Z \to C$

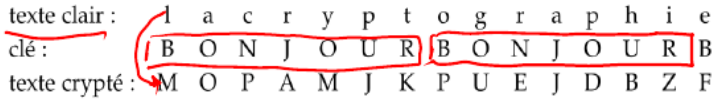
Cryptage facile à faire mais élémentaire à briser de façon brute-force (il n'y a que 26 décalages possibles ; tous les essayer se fait très rapidement).

6.3 Substitution monoalphabétique

On définit une table de remplacement unique (chaque lettre claire a une et une seule lettre codée). Cette propriété garantit le décodage unique. Briser ce code est un peu plus difficile ; on peut chercher dans un texte codé la lettre la plus fréquente, qui correspond sûrement à un e (la lettre la plus fréquente en français). On peut aussi chercher des mots en rapport (si c'est commercial chercher le mot *vendre*, ou *action*). L'attaque exhaustive est longue à la main (26! possibilités), mais peut se faire rapidement avec un ordinateur.

6.4 Chiffre de Vigenère

Cryptage symétrique, substitution polyalphabétique. On choisit un mot/phrase. On associe à chaque lettre un chiffre (A=0, Z=25); on aligne ensuite P avec des répétitions de K, et on décale la position des lettres de P d'autant. En calcul, on peut simplement faire $(P + K) \mod 26$
Par exemple, avec la clé BONJOUR (associée aux chiffres 1,14,13,914,20,17):



Il y a 26^n clés (avec n la taille de K)→ attaque exhaustive difficile. Si la clé est courte et le texte est long, il est possible de faire des attaques basées sur la la fréquence des lettres.

6.5 Chiffre de Vernam

Cryptage symétrique. On parle d'un masque *jetable*, ou *à usage unique*(one time pad). On doit avoir K et P de même longueur. On calcul ensuite C par $\boxed{P \bigoplus K}$ (xor, défini parle tableau de la Figure 12) . Il faut la clé et le texte soient indépendants (par exemple faire un lancer de dés pour déterminer K)
Le décodage se fait par $\boxed{P = C \bigoplus K}$. Ce système est a confidentialité parfaite (pour autant que P et K sont indépendants)

xor	0	1
0	0	1
1	1	0

Figure 12: La table du xor

Exemple 22. Soit notre clé K = 1101 et notre texte P = 0100. C se trouve de cette manière : $\begin{array}{rr} & 0100 \\ \bigoplus & 1101 \\ \hline = & 1001 \end{array}$ Donc $C = 1001$. Le décryptage se fait de la même manière :

$$\begin{array}{rr} & 1001 \\ \bigoplus & 1101 \\ \hline = & 0100 \end{array}$$ et on retrouve P

6.6 La confidentialité parfaite

Un système est à confidentialité parfaite si connaître C ne dit absolument rien sur P (nous supposons que l'intrus connaît la méthode de chiffrement utilisée).

Exemple 23. Prendre P et K deux chiffres de 1 à 7, et les multiplier pour donner C n'est pas à confidentialité parfaite : Si $C = 49$, alors $P = K = 7$

Par le **Théorème 6.2** : Nous prenons un texte à confidentialité parfaite. Si la clé et le texte clair sont choisit indépendamment, alors

$$H(P) \leq H(C) \leq H(K)$$

7 Chapitre 7 : Arithmétique Modulaire

7.1 Division Euclidienne

Pour tout a, b, des entiers, il existe un couple q, r tels que $a = bq + r$ (avec q = quotient et r = reste et $0 \leq r \leq |b| - 1$).

Exemple 24. Pour $a = 23$ et $b = 5$, alors $q = 4,\ r = 3$, car $23 = 5 \cdot 4 + 3$.

Notons que l'on peut calculer (si $b > 0$) : $q = \lfloor \frac{a}{b} \rfloor,\ r = a - bq$. On dénote r comme étant $a \mod b$ (reste du modulo-division entière de a par b)

7.2 La congruence modulo

Deux entiers a et b sont "congru modulo m" si ils ont le même reste dans la division par m, et l'on écrit

$$a \equiv b \pmod{m}$$

Exemple 25. $33 \equiv 23 \equiv 13 \equiv 3 \pmod{10}$

- On dit que x est divisible par m si $x \equiv 0(\mod m)$
- Tout nombre est congru modulo m à son reste dans la division par m
- En général, $x \equiv y(\mod a) \iff (x-y) \equiv 0(\mod a)$

7.2.1 Relation d'équivalence

La congruence modulo est une relation d'équivalence sur $\mathbb{Z}$, donc :

- Réflexivité : $a \equiv a(\mod m)$
- Symétrie : $a \equiv b(\mod m) \iff b \equiv a(\mod m)$
- Transitivité : Si $a \equiv b(\mod m)$ et $b \equiv c(\mod m)$ alors $a \equiv c(\mod m)$

De plus, par le théorème de l'arithmétique modulaire : Si $a \equiv a'(\mod m)$ et $b \equiv b'(\mod m)$, alors

$$\begin{aligned} a+b &\equiv a'+b'(\mod m) \\ ab &\equiv a'b'(\mod m) \\ a^n &\equiv a'^n(\mod m) \end{aligned}$$

7.2.2 Calculs/trucs

Il est depuis là facile de faire des calculs ; par exemple, 2^{1000} est-il divisible par 3 ? Nous savons que $2^{1000} \equiv (-1)^{1000} \equiv 1 \mod 3$. Donc ça n'est pas divisible par 3.

De même, $37^{9876543}$ est il multiple de 7 ? $37^{9876543} \mod 7 = 2^{9876543} \mod 7 = (2^3)^{\frac{9876543}{3}} \mod 7 = 8^{3292181} \mod 7 = 1^{3292181} \mod 7 = 1 \mod 7 = 1$

Il est aussi facile de calculer le reste de la division par 2 : Si le dernier chiffre est pair le reste est de 0 ; en revanche, il est de 1 si le chiffre est impair.

Finalement, le calcul du reste de la division par 9 se fait en divisant *la somme des chiffres qui composent le nombre* par 9. Ainsi, $1234567890 \mod 9 = 1+2+3+4+5+6+7+8+9+0 \mod 9 = 45 \mod 9 = 4+5 \mod 9 = 9 \mod 9 = 0$

Notons aussi que les calculs se font normalement aussi sur les modulo. Seule la division ne se fait pas normalement. Pour ça, on peut changer "mod" en "bq + r", afin de pouvoir faire les calculs normaux.

7.3 Procédure MOD 97-10

Utilisée essentiellement par les banques, comme numéro de contrôle ; une erreur courante est d'inverser 2 chiffres, donc ajouter ce chiffre de contrôle permet de repérer l'immense majorité des erreurs.

Pour ce faire, il faut prendre notre chiffre bancaire, ajouter 00 à la fin, calculer le r = numéro mod 97. Ensuite, remplacer le 00 par $\boxed{98\text{-}r}$[1]. Ce nouveau chiffre doit être congru à 1 modulo 97.

Notons que dans les numéros IBAN, nous avons 2 lettres et 2 chiffres. Les 2 lettres sont remplacées par un numéro à 4 chiffres ($A \to 10, ..., Z \to 35$). On met ce numéro (et les 2 chiffres de contrôle) à la fin, et le calcul se fait là-dessus.

En gros :

1. Ajouter 00 à la fin
2. r = reste de la division par 97
3. Chiffre de contrôle $c = 98 - r$; remplacer 00 par c
4. Vérification : le mot reçu doit être $\equiv 1 (\text{mod } 97)$

Exemple 26.

1. $x = 21235123400$
2. $21235123400 \equiv 91 \pmod{97} \to r = 91$
3. $c = 98 - r = 98 - 91 = 07$
4. Le chiffre avec chiffres de contrôle : 21235123407

7.4 Décomposition en nombres premiers

- Premier théorème : Pour tout entier a positif, il existe une suite d'entiers $p_1 < p_2 < ... < p_k$ et une suite unique d'exposants $\alpha_1 > 0, ..., \alpha_k > 0$ tels que

$$a = p_1^{\alpha_1} \cdot ... \cdot p_k^{\alpha_k}$$

  Les nombres $p_1, ..., p_k$ sont appelés les facteurs premiers.

- Un entier ($\in \mathbb{Z}$) est premier s'il n'est divisible que par 1 et lui-même.
- Soient 2 entiers a et b. a divise b si et seulement si tous les facteurs premiers de a apparaissent dans la décomposition de b, à une puissance égale ou supérieure.
- Le PGCD : on décompose les deux chiffres, et on prend tout ce qui est en commun ; si un facteur apparaît 2x, on le prend à la plus petite puissance (car cette puissance est en commun).
- Deux nombres sont premiers entre eux si et seulement si leur pgcd est de 1 (aucun facteur premier en commun)
- Deux nombres premiers sont premiers entre eux.
- Pour un entier a et un premier p tels que $1 \leq a \leq p - 1$, alors a et p sont premiers entre eux
- Soient a et b deux entiers premiers entre eux, et c un entier. Si a et b divisent c, alors ab divise c.

[1] 98 et non pas 97, afin que le mod du nouveau chiffre fasse 1 et pas 0

8 Arithmétique modulaire Z/mZ

8.1 Classe de congruence

Pour $m \geq 2$ (le module) un entier fixé. Pour tout entier a, on appelle classe de congruence de a modulo m ($[a]_m$) tous les entiers a' tels que $a \equiv a'(\mod m)$. L'ensemble des classe de congruence modulo m s'écrit $\mathbb{Z}/m\mathbb{Z}$

La somme et le produit se font instinctivement : $[a]_m + [b]_m = [a+b]_m, \quad [a]_m \cdot [b]_m = [ab]_m$ L'addition se fait normalement (transitivité, commutativité, associativité, élément neutre et symétrique). La multiplication aussi, sauf pour l'élément inverse.

8.2 Notation

- $k[a]_m = [k]_m[a]_m = [ka]_m$
- $-[a]_m = [-a]_m = [-a+km]_m$

8.3 L'inverse

$[a]_m$ possède $[b]_m$ comme inverse si $[a]_m[b]_m = [1]_m$. L'inverse est alors noté $([a]_m)^{-1}$
Attention ! Il est possible qu'un entier n'ait pas d'inverse. Par exemple, $[2]_4$ n'est pas inversible.

8.3.1 Remarques

- $[0]_m$ n'a jamais d'inverse
- S'il existe, l'inverse est unique
- $([a]_m)^k = \overbrace{[a]_m \cdot [a]_m \cdot \ldots \cdot [a]_m}^{k\times} = [a^k]_m$
- Pour p un premier, tous les nombres de $\mathbb{Z}/p\mathbb{Z}$ sont inversibles (sauf $[0]_p$ bien sûr)
- Selon Euclide et Bézout, $[a]_m$ est inversible si et seulement si a et m sont premiers entre eux.

8.4 Algorithme d'Euclide

Soient a et b deux entiers, avec $b \neq 0$, et soit $a = bq + r$. Alors

$$pgcd(a,b) = pgcd(b,r)$$

Exemple :
$pgcd(122,22) = pgcd(22,12) = pgcd(12,10) = pgcd(10,2) = pgcd(2,0) = 2$

8.5 L'identité de Bézout

Soient a et b deux entiers ; il existe u et v entiers tels que $au + bv = pgcd(a,b)$

8.6 Inverser un chiffre

L'algorithme ci-dessous permet de calculer l'inverse d'un nombre dans une base donnée. Difficile de le faire en mots, le mieux est avec un exemple :

Exemple 27. Cherchons l'inverse de 9 dans la base 95 :
$\boxed{95} = 10 \cdot \boxed{9} + 5 \iff \boxed{95} - 10 \cdot \boxed{9} = 5 \curvearrowright$
$\boxed{9} = 5 + 4 \iff \boxed{9} - 5 = 4 = \boxed{9} - (\boxed{95} - 10 \cdot \boxed{9}) = 11 \cdot \boxed{9} - \boxed{95} = 4$
$5 = 4 + 1 \iff 5 - 4 = 1 = \Big(\boxed{95} - 10 \cdot \boxed{9}\Big) - \Big(11 \cdot \boxed{9} - \boxed{95}\Big) = 2 \cdot \boxed{95} - 21 \cdot \boxed{9}$
Donc nous avons montré que

$$1 = 2 \cdot 95 - 21 \cdot 9 = \underbrace{[2 \cdot 95]_{95}}_{0} - [21 \cdot 9]_{95}$$

et ainsi également que

$$[-21]_{95} \cdot [9]_{95} = [1]_{95}$$

et donc l'inverse de 9 est logiquement

$$[-21]_{95} = [95 - 21]_{95} = [74]_{95}$$

9 Chapitre 9 : Algèbre abstraite

9.1 Notation

- e = élément neutre
- $\varphi(m)$ = indice d'Euler (de m)
- $\star$ une opération, définie mais quelconque.

9.2 Groupe commutatif

Soit $(G, \star)$ un ensemble G muni de l'opération binaire $\star$. C'est à dire un mécanisme qui associe à deux éléments a et b de G (distincts ou non) un élément de G noté $a \star b$.

$(G, \star)$ est appelé un groupe commutatif (ou groupe abélien) s'il possède les propriétés :

- Associativité $a \star (b \star c) = (a \star b) \star c$
- Neutre il existe un e tel que $a \star e = e \star a = a$
- Symétrique pour tout élément a, il existe un a' tel que $a \star a' = e$
- Commutativité $a \star b = b \star a$

9.2.1 Groupe modulaire multiplicatif

On connaît déjà $\mathbb{Z}/m\mathbb{Z}$ qui représente tous les entiers de classe m.

Maintenant nous considérons $\mathbb{Z}/m\mathbb{Z}^*$, qui est l'ensemble des éléments inversibles de $\mathbb{Z}/m\mathbb{Z}$ (donc ceux qui sont plus petits que m, et premier avec lui).

9.3 Produit cartésien

$(\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/5\mathbb{Z})$ donnera $(00), (01), (02), (03), (04), (10), (11), (12), (13), (14)$. On calcule symbole par symbole (donc le second se calculera mod 5 alors que le premier se calculera mod 2). On peut faire autant de multiplications qu'on veut. Si $(G, \star)$ et $(H.\star)$ sont des groupes, alors $(G \times H, \star)$ est aussi un groupe.

9.4 Isomorphisme

Deux groupes sont isomorphes si on peut renommer les éléments de manière à ce que les tableaux correspondent.

9.5 Période d'un élément

Soit $(G, \star)$ un groupe commutatif fini (doté d'un élément neutre).

1. Pour tout élément $a \in G$, il existe un entier $k \geq 1$ tel que $\overbrace{a \star a \star \ldots \star a}^{k\times} = e$. Le plus petit de ces entiers est appelé la période de a.
2. Pour tout entier positif l, $\overbrace{a \star a \star a \star \ldots \star a}^{l \text{ fois}} = e$ si et seulement si la période divise l ; autrement dit, la période de chaque éléments de G divise le nombre d'éléments du groupe.

Exemple 28.
Dans $(\mathbb{Z}/12\mathbb{Z}, +)$, la période est le plus petit élément entier positif k tel que $k[a]_{12} = [0]_{12}$
Dans $(\mathbb{Z}/12\mathbb{Z}, \cdot)$, la période est le plus petit élément entier positif k tel que $([a]_{12})^k = [1]_{12}$

Théorème 9.3 (Lagrange) Soit $(G, \star)$ un groupe commutatif fini, de cardinal n. La période de tout élément de G divise n.

En particulier, notons e l'élément neutre de G. Pour tout $a \in G$:

$$\overbrace{a \star a \star a \star \ldots \star a}^{n\times} = e$$

De ce théorème important découle ce corollaire :
Corollaire 9.4 (Théorème d'Euler. Nous posons $\varphi(m)$ (l'indicatrice d'Euler), qui est le nombre d'éléments (le cardinal) de $\mathbb{Z}/m\mathbb{Z}^*$. Alors, pour tout entier positif m et tout entier a premier avec m :

$$([a]_m)^{\varphi(m)} = [1]_m$$

ou encore

$$a^{\varphi(m)} \equiv 1 (\text{mod } m)$$

9.6 Restes chinois, l'application ψ

Tiré du jeu des restes chinois : on remplit par la diagonale une boite torique[2]. Dans certains cas la boite se remplit sans problème, dans d'autres elle "bouclera" (des cases seront remplies plusieurs fois, d'autres jamais). Mathématiquement, cela revient à calculer le modulo selon le nombre de ligne et de colonnes et les placer aux coordonnées indiquées.

	0	1	2
0	0	4	8
1	9	1	5
2	6	10	2
3	3	7	11

$m_1 = 3, m_2 = 4$

Figure 13: Remplissage d'une boîte de 3×4

Exemple 29. Exemple : Avec m_1 le nombre de colonnes et m_2 le nombre de lignes, placer un chiffre k dans la boite le place aux coordonnées $[k]_{m_1}, [k]_{m_2}$. Dans la boite de la Figure 13 7 ira à la position $([7]_3, [7]_4) = (1, 3)$, donc à la colonne 1, ligne 3

Cela nous permet de comprendre un point essentiel : Cette boîte se remplira complètement (sans bouclage) si et seulement si les entiers m_1 et m_2 sont premiers entre eux.

9.6.1 Le théorème

L'application ψ est définie comme telle :

$$\psi : \begin{cases} \mathbb{Z}/m_1m_2\mathbb{Z} \to \mathbb{Z}/m_2\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \\ [k]_{m_1m_2} \to ([k]_{m_1}, [k]_{m_2}) \end{cases}$$

Si m_1, m_2 sont premiers entre eux, l'application est bijective et représente un isomorphisme pour $\cdot$ et $+$ (donc entre $(\mathbb{Z}/m_1m_2\mathbb{Z}, +)$ et $(\mathbb{Z}/m_2\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z}, +)$ par exemple)

Mais si ces deux chiffres ne sont pas premiers entre eux, alors l'application n'est ni injective ni surjective, car des cases seront vides alors que d'autres auront plusieurs chiffres (principe des tiroirs)

10 Chapitre 10 : RSA

10.1 La base

- On prend deux premiers secrets p et q (et un exposant publique e)
- La clé publique $\boxed{K = m = pq}$
- La clé privée $\boxed{k = ppmc\big((p-1),(q-1)\big)}$
- On change le texte clair en éléments de $\mathbb{Z}/m\mathbb{Z} \to \boxed{C = P^e}$
- e est l'exposant de chiffrement (publique) premier avec k
- L'exposant de déchiffrement f est tel que : $\boxed{[f]_k = ([e]_k)^{-1}}$
- On déchiffre de la même manière : $P = ([C]_m)^f$

[2] une fois sorti de la boite, on continue à la manière d'une sphère, donc en haut si on finit en bas, à gauche si on finit à droite

11 Chapitre 10 : Les codes correcteurs

11.1 Définitions

- A := Alphabet : Tous les symboles qui composent notre code
- C := ensemble de mots de code $C \subset A^n$
- n := longueur commune à tous les mots de code.
- $k = \log_{card(A)}(card(C)) \simeq$ longueur des mots sans code si on les décrit avec l'alphabet A
- $r = \frac{k}{n}$:= rendement = proportion de bits originaux.

11.2 Distance de Hamming

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

$$d(x, y) \stackrel{\text{def}}{=} card\{i \in \{1, \ldots, n\} : x_i \neq y_i\}$$

Si les longueurs de x et y ne sont pas les mêmes, la longueur n'est pas définie.

11.2.1 La distance minimale

La distance minimale est la plus petite distance dans un ensemble de mots de code. Donné un ensemble C, on peut comparer toutes les distances de Hamming, et la plus petite est la distance minimale. Il faut cependant que les deux chaînes soient différentes.

11.2.2 Le cas d'un code linéaire

Notons que pour un code linéaire (présence du mot nul et chaque somme de deux mots est un mot), la distance minimale revient à regarder le nombre de symboles différents de 0. (uniquement pour les codes linéaires !!)

11.3 Modèles de Canal

- Canal à effacement : un ou plusieurs symboles sont effacés, mais on sait où. le poids de l'effacement correspond au nombre de symboles effacés. : 0100111 $\rightarrow$ 0?001?1
- Canal à erreurs : un ou plusieurs symboles sont modifiés (sans comparer on ne sait pas lesquels sont modifiés) 0100111 $\rightarrow$ 0000101 (le second et l'avant dernier symbole ont été modifiés, mais le code arrivant n'est pas forcément faux).

11.4 Décodeurs

Un décodeur à distance minimal : Donné un mot de code (pas forcément dans notre ensemble C), le décodeur à distance minimale prend le mot de code dans C avec la plus petite distance.

Détection d'Effacement : Par définition de l'effacement, on est capable de détecter tous les effacements.

Détection d'Erreurs : Le décodeur d'un code C est capable de détecter toutes les erreurs de poids $\leq p$ si et seulement si $p < d_{min}(C)$

Correction d'Effacement : Le décodeur d'un code C est capable de corriger tous les effacements de poids $\leq p$ si et seulement si $p < d_{min}(C)$

Correction d'Erreurs : Le décodeur d'un code C est capable de corriger toutes les erreurs de poids $\leq p$ si et seulement si $p < \frac{d_{min}(C)}{2}$

11.5 Borne de Singleton

Pour un code en bloc C de longueur n et de rendement r, la distance minimale satisfait

$$d_{min}(C) \leq n(1-r)+1$$
$$d_{min}(C) \leq n-k+1$$

12 Chapitre 12 : Corps finis et espaces vectoriels

On connaissait $(K, \star)$ un groupe commutatif, on parle maintenant de $(K, +, \cdot)$, un corps commutatif. C'est un corps commutatif si $(K, +)$ est un groupe commutatif, si tous les éléments sauf 0 sont inversibles, si $(K^*, \cdot)$ est un groupe commutatif, et si la distributivité est respectée $\big(a \cdot (x+y) = a \cdot x + a \cdot y\big)$

Dans un corps fini, la période pour l'addition de 1 (l'élément neutre pour la multiplication) s'appelle la caractéristique du corps (toujours un nombre premier). La caractéristique de $\mathbb{Z}/p\mathbb{Z}$ est p (quand p est premier bien sûr).

Théorème :

1. Le cardinal d'un corps fini est une puissance de sa caractéristique
2. Tous les corps finis de même cardinal sont isomorphes
3. Pour tout nombre premier p et tout entier $m \geq 1$, il existe un corps fini de cardinal p^m

12.1 Le corps F_4

On avait déjà vu le groupe E_4, qui comprend tous les éléments modulo 4. L'addition et multiplication sont définis normalement, simplement que $3+3 = 6 = 2$. Maintenant, on a F_{p^m} seulement pour p premier et m entier. Par exemple, $F_4 = F_{2^2}$ On aura donc un groupe avec 4 éléments, mais répondant au modulo 2. Les tables de ce corps sont à droite. L'opération est une bijection ; on s'aide de cette information pour remplir le tableau (évident que $0 + x = x$, et que $x + x = x(1+1) = x0 = 0$, mais difficile avec $a + b$). Les éléments se remplissent assez logiquement.

+	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

·	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

Figure 14: La table de F_4

12.1.1 Petit truc

Figure 15: Les tables de F_4 et de $\mathbb{Z}/2\mathbb{Z}^2$

+	0	1	a	b
0	0	1	a	b
1	1	0	b	a
a	a	b	0	1
b	b	a	1	0

+	00	11	01	10
00	00	11	01	10
11	11	00	10	01
01	01	10	00	11
10	10	01	11	00

·	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a	b	1
b	0	b	1	a

·	00	11	01	10
00	00	00	00	00
11	00	11	01	10
01	00	01	10	11
10	00	10	11	01

Pour l'addition, on peut tout a fait considérer F_4 comme $\mathbb{Z}/2\mathbb{Z}^2$ (deux éléments sur un modulo 2) ; mais la multiplication diffère un peu : si l'addition se comporte comme une addition standard (xor, sans retenue) sur deux bits, et la multiplication se comporte bizarrement : 00 met a 00, 11 ne change rien, et $01 \times 01 = 10$, $01 \times 10 = 11$ et $10 \times 10 = 01$. Donc attention !

Le tableau complet se trouve ci-contre, à la Figure 15

12.2 (sous-)Ensemble vectoriel

Soit V un sous ensemble vectoriel de $F_{p^m}^k$; il est engendré par plusieurs vecteurs de longueur k. Pour trouver la dimension du sous espace, on met les vecteurs dans une matrice et on réduit (attention à ne pas diviser) ; le nombre de colonnes pivots donne la dimension de V, le cardinal s'obtient avec $(p^m)^{dim(V)}$ (donc dimension 3 dans F_{13} nous donnera un cardinal de 13^3). On peut nous demander de trouver p, le nombre minimal d'équations pour engendrer V. Pour cela on reprend notre matrice réduite et on lui met comme solution 0. Ensuite on passe les éléments de l'autre côté, pour trouver la forme paramétrique. Attention, par le théorème du rang, il faut que $p + dim(V) = k$, donc que la dimension + le nombre de solutions = le nombre de variables/colonnes.

Résoudre les équations et réduire les matrices dans ces corps se fait d'une autre manière qu'on a l'habitude. Le but est toujours le même, mais au lieu de diviser on peut multiplier (dans F_{13}, pour "diviser" 8 par lui-même, il suffit de multiplier par 5)

12.3 Espaces vectoriels

Notons la **Définition 12.3** des espaces vectoriels, importante :

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

- Associativité pour la multiplication scalaire : $\lambda(\mu\vec{v}) = (\lambda\mu)\vec{v}$
- Identité : $1 \cdot \vec{v} = \vec{v}$
- Distributivité : $\lambda(\vec{u} + \vec{v}) = \lambda\vec{u} + \lambda\vec{v}$ et $(\lambda + \mu)\vec{v} = \lambda\vec{v} + \mu\vec{v}$

12.3.1 Sous-espace vectoriel

Un sous-ensemble $\mathcal{S}$ de $\mathcal{V}$ ($\mathcal{S} \subset \mathcal{V}$) est un sous-espace vectoriel s'il est aussi un espace vectoriel sur $\mathcal{K}$. Cela revient à s'assurer que deux opérations sont définies :

$$\lambda\vec{u} \in \mathcal{S} \text{ et } \vec{u} + \vec{v} \in \mathcal{S} \qquad \forall\lambda \in \mathcal{K},\ \vec{u}, \vec{v} \in \mathcal{S}$$

Exemple 30. Considérons l'espace vectoriel $\mathbb{F}_7^3$ (donc l'ensemble des vecteurs représentés à la figure 16) ainsi que l'ensemble $\mathcal{S}$ des vecteurs de la forme $(u, 3u, 6u)$. C'est un sous-espace vectoriel car

$$\lambda(u, 3u, 6u) = \big(\lambda u, 3(\lambda u), 6(\lambda u)\big) \in \mathcal{S}$$
$$(u, 3u, 6u) + (v, 3v, 6v) = \big(u + v, 3(u + v)6(u + v)\big) \in \mathcal{S}$$

Nous pouvons aussi considérer le sous-espace vectoriel $\mathcal{S}$' des vecteurs $\vec{x} = (x_1, x_2, x_3)$ qui satisfont $x_1 + 4x_2 + 3x_3 = 0$ (c.f. exemple 12.5 du livre pour la preuve)

Notons également ces quelques termes : Une combinaison linéaire de vecteurs $\vec{v_i} \in \mathcal{V},\ i = 1 \dots m$ est une somme de la forme

$$\vec{u} = \sum_{i=1}^{m} \lambda_i \vec{u}_i$$

où les cœfficients λ_i sont des scalaires. L'ensemble des vecteurs $\vec{u}$ engendrés par toutes les combinaisons linéaires de m vecteurs v_1 forme un sous espace vectoriel, appelé le sous-espace vectoriel engendré par les $\vec{v_i}$.

Il sera parfois nécessaire de vérifier que des vecteurs sont linéairement indépendants. Pour cela il suffit de réduire la matrice des vecteurs (selon la méthode du pivot de Gauss, vu en algèbre linéaire) ; les lignes non-nulles sont alors indépendantes entre elles. Ces vecteurs forment une base de $\mathcal{V}$. Cela signifie que l'ensemble des vecteurs de $\mathcal{V}$ peut s'écrire de manière unique comme une combinaison de ces vecteurs de la base ; les coefficients d'une telle combinaison s'appellent les coordonnées du vecteur relativement à la base

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 6 \\ 0 & 1 & 0 \\ \vdots & \vdots & \vdots \\ 6 & 6 & 6 \end{pmatrix}$$

Figure 16: Le corps $\mathbb{F}_7^3$

13 Chapitre 13 : Les Codes Linéaires

Pour les exemples, le code C de la Figure 17 sera utilisé.
Définition 13.1 :Soit C un code en bloc de longueur n. On dit que C est un code linéaire si :

1. L'alphabet du code est un corps fini K
2. Le code est un sous-espace vectoriel de K^n

Cela ce traduit par la vérification de deux items (outre que l'alphabet est un corps fini) :

- Si la somme de deux mots de code est encore un mot de code
- Si le vecteur nul $(\vec{0})$ est bien un mot de code

Exemple 31. Est-ce que le code C est linéaire ?

1. L'alphabet est F_2, ce qui est un corps (fini)
2. $v_4 = v_1 + v_2,\ v_5 = v_1 + v_3,\ v_6 = v_2 + v_3,\ v_7 = v_1 + v_2 + v_3$
   donc C est l'ensemble des 8 combinaisons linéaires de $v_1, v_2, v_3 \rightarrow$ OUI, c'est un code linéaire.

On peut repérer si un code n'est pas linéaire par inspection (absence du vecteur nul ou si la somme de deux vecteurs n'est pas un mot), ou par taille : il est nécessaire qu'un code D-aire possède D^k mots de code, pour un k entier. Si un code binaire possède 3,5,6,... mots de code, c'est impossible, De même un code ternaire doit posséder 3,9,27,... mots de code.

13.1 Dimension et distance minimale

Soit d = dimension du code linéaire (= la dimension du sous-espace vectoriel des mots de code). Il y a 2^d mots de code, donc $k = d$ (k un des paramètres du code, c.f. Figure 17). **Définition/Théorème 13.2 :** Si K est un corps fini, le poids de Hamming de $\vec{x} \in K^n$ est le nombre de composantes non-nulles, c'est à dire aussi $w(\vec{x}) \stackrel{\text{def}}{=} d(0, \vec{x})$.
La distance minimale d'un code linéaire C est égale à

$$d_{min}(C) = \min_{\substack{\vec{x} \in C \\ \vec{x} \neq \vec{0}}} w(\vec{x})$$

$k = 3$	$n = 7$
000	$\vec{v}_0 =$ 0000000
001	$\vec{v}_1 =$ 0011100
010	$\vec{v}_2 =$ 0111011
100	$\vec{v}_3 =$ 1110100
011	$\vec{v}_4 =$ 0100111
101	$\vec{v}_5 =$ 1101000
110	$\vec{v}_6 =$ 1001111
111	$\vec{v}_7 =$ 1010011

Figure 17: Code C d'exemple

Exemple 32. Pour le code C, il y a 8 poids à calculer : $0, 3, 3, 4, 4, 3, 5, 5$ (le nombre de "1" de chaque mot). Le minimum (outre 0) est 3, donc $d_{min}(C) = 3$

Notons bien que la distance à 0 est indépendante de la valeur non nulle de la coordonnée.

Exemple 33. Pour un code dans $\mathbb{F}_7^3$, le mot 001 a une distance de 1, et le mot 111 a une distance de 3. Mais attention, car le mot 666 a aussi une distance de 3 et le mot 050 est de distance 1. On ne regarde pas la valeur de la composant, mais bien si elle est nulle ou pas.

13.2 Matrice Génératrice

Définition 13.3 Soit C un code linéaire sur le corps K, de longueur n et dimension k. Soit $(\vec{v_1}, \ldots, \vec{v_k})$ une base de C. La matrice obtenue en écrivant à la i-ième ligne le vecteur $\vec{v_i}$ est appelée une matrice génératrice du code.

En gros, prendre la base (nombre minimal de vecteurs indépendants qui engendrent tous les autres mots) et les mettre dans une matrice.

Exemple 34. On a vu que la base de C est constituée des vecteurs (v_1, v_2, v_3). Donc la matrice génératrice

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

Cette matrice génératrice donne tous les mots de code. Donné un mot (pas encodé) $\vec{u}$, son encodage se trouve par $\vec{x} = \vec{u} \cdot G$

Exemple 35.

$$\vec{u} = (1,0,1) \to \vec{x} = \vec{u}G = (1,0,1) \cdot \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} = 1,1,0,1,0,0,0$$

Ce qui correspond au mot encodé, selon notre tableau (Figure 17)

13.3 Forme Systématique

La forme systématique d'une matrice est simplement sa forme échelonnée réduite. Le premier chiffre non nul de chaque ligne doit être un 1, et tous les éléments dans la colonne au dessus et au dessous de ce 1 doivent être nuls.

Exemple 36. Les deux matrices suivantes sont systématiques :

$$\begin{pmatrix} 1001111 \\ 0100111 \\ 0011100 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

13.4 Matrice de contrôle

Le but de la matrice de contrôle est de vérifier qu'un séquence encodée soit valide, existe. Il faut donc réduire notre matrice G à une forme échelonnée (forme systématique) et exprimer les variables dépendantes telles que la somme vaut 0 ; il faut que $\vec{x}H^T = \vec{0}$

Exemple 37. Donnée la matrice génératrice $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$ dans F_5. Nous la réduisons comme nous savons le faire jusqu'à une forme symétrique : $\begin{pmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$
Le code permet d'encoder des mots de 3 symboles $u_1u_2u_3$ à l'aide des formules (regarder les colonnes) : $\begin{cases} x_1 = u_1 \\ x_2 = u_2 \\ x_3 = u_3 \\ x_4 = 3u_1 + 3u_2 \\ x_5 = 2u_1 + 4u_2 \end{cases}$
On prend nos deux dernières équations, et on les égalise à 0 : $\begin{cases} x_4 - 3x_1 - 3x_2 = 0 \\ x_5 - 2x_1 - 4x_2 = 0 \end{cases} \rightarrow \begin{cases} 2x_1 + 2x_2 + x_4 = 0 \\ 3x_1 + x_2 + x_5 = 0 \end{cases}$ et il reste plus qu'à mettre en vecteurs et transposer :

$$(x_1, x_2, x_3, x_4, x_5) \cdot H^T = (\mathbf{x}) \cdot \underbrace{\begin{pmatrix} 2 & 3 \\ 2 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{H^T} = (0, 0)$$

Notons ainsi le théorème 13.1 :

Théorème 13.1 *Si la matrice génératrice G est sous la forme systématique* $G = [I_k \quad P]$ *alors une matrice de contrôle est*

$$H = [-P^T \quad I_{n-k}]. \tag{13.11}$$

$G_3 = \begin{pmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ est une matrice génératrice du code

$H = \begin{pmatrix} -3 & -3 & 0 & 1 & 0 \\ -2 & -4 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 \end{pmatrix}$ est une matrice de contrôle

Figure 18: Théorème 13.1 sur les matrices de contrôle

13.5 Syndrome

Soit $\vec{x} \in K^n$ un mot de n symboles (pas nécessairement un mot de code). Alors

$$\vec{e} = \vec{x}H^T$$

s'appelle le **syndrome**. Cela implique que

$$\vec{x} \text{ est un mot de code } \iff \vec{e} = \vec{x}H^T = \vec{0}$$

par définition de H.

14 Chapitre 13 : Les Codes de Reed-Solomon

Les codes de Reed-Solomon sont très utiles ; c'est le code le plus "efficace" pour encoder de l'information. Le code est linéaire (donc sur F_{p^q}, p premier et q entier positif). Le rendement est variable ($r = \frac{k}{n}$ et on choisir k, n tels que $k \leq n \leq p^q$). Ces codes atteignent la borne de Singleton (ils sont MDS) par définition

14.1 Polynôme

Prenons $\vec{u} = (u_1, u_2, \ldots, u_k) \in F^k$ pour certains champs finis F. Chaque $\vec{u}$ définit un polynôme $P_{\vec{u}}(x)$ (dans F) :

$$P_{\vec{u}}(x) = u_1 + u_2 x + u_3 x^2 + u_4 x^3 + \ldots + u_k x^{k-1} \tag{8}$$

$P_{\vec{u}}(x)$ peut alors être évalué en tout $x \in F$
Le **degré** du polynôme est la plus grande puissance de x avec un coefficient non-nul.

Exemple 38. Soit $F = F_5$ e $\vec{u} = (2, 4, 3)$. $P_{\vec{u}}(x) = 2 + 4x + 3x^2$ et est de degré 2. Notons que $P(x) = 2$ (par exemple) est de degré 0 et que le degré de $P(x) = 0$ n'est pas défini (ou $-\infty$).

Nous savons, par le théorème 14.1 qu'un polynôme de degré $k-1$ (comme à l'équation (8)) a au maximum $k-1$ solutions. S'il a k solutions ou plus, alors $\vec{u} = \vec{0}$.

Exemple 39. $x^2 - 3x + 2$ est de degré k-1 = 3-1 = 2, et a exactement 2 solutions : $x = 1$ et $x = 2$.
En revanche, $x^2 + 3x + 2$ n'a aucune solution.

14.2 Construction du code

La marche à suivre officielle se présente comme suit :

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

1. L'alphabet est un corps fini K de cardinal $\geq n$
2. Choisissons n éléments distincts de $K, a_1, a_2, a_3, ..., a_n$. Une suite de k symboles $\vec{u} = (u_1, u_2, ..., u_k) \in K^k$ est encodée en la suite de n symboles $\vec{x} = (x_1, ..., x_n) \in K^k$ définie par

$$x_i = P_{\vec{u}}(a_i) \quad \text{pour } i = 1, .., n$$

Le code de Reed Solomon C est l'ensemble de tous les encodages possibles, pour tous les $\vec{u} \in K^k$. C'est donc un code en bloc de longueur n.

Exemple 40. **En résumé :** On veut construire un code sur F_5. On doit choisir un n entre 1 et 5, et un k entre 1 et n. Prenons $n = 5$ (maximum) et $k = 2$. Nous devons maintenant choisir 5 a_i. Comme nous devons choisir 5 éléments distincts dans F_5 nous n'avons pas le choix : $a_1 = 0$, $a_2 = 1, ..., a_5 = 4$. Il faut maintenant encoder : on choisit tous les $\vec{u}$ (de longueur 2) possibles dans F_5. Par exemple, pour $\vec{u} = (3, 2)$, alors $P_{\vec{u}}(X) = 3 + 2X$. Donc $P_{\vec{u}}(a_i) = 3 + 2 \cdot a_i$. Toujours pour $\vec{u} = (3, 2)$, nous obtenons

$$P_{\vec{u}}(a_1) = P_{\vec{u}}(0) = 3$$
$$P_{\vec{u}}(a_2) = P_{\vec{u}}(1) = 0$$
$$P_{\vec{u}}(a_3) = P_{\vec{u}}(2) = 2$$
$$P_{\vec{u}}(a_4) = P_{\vec{u}}(3) = 4$$
$$P_{\vec{u}}(a_1) = P_{\vec{u}}(4) = 1$$

et donc $\vec{x} = (3, 0, 2, 4, 1)$. Il faut répéter l'opération pour les n^k suites.

14.3 La matrice génératrice

Un autre moyen de construire tous les mots de code est de passer par la matrice génératrice G. Pour cela, nous devons prendre une base de C, et donc k vecteurs $\vec{u}$ linéairement indépendants. La manière la plus simple de créer G est de choisir des $\vec{u}$ qui formeront la I_k. Il ne reste alors plu qu'à créer G selon ce tableau

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ a_1 & a_2 & a_3 & ... & a_n \\ (a_1)^2 & (a_2)^2 & (a_3)^2 & ... & (a_n)^2 \\ ... & ... & ... & ... & ... \\ (a_1)^{k-1} & (a_2)^{k-1} & (a_3)^{k-1} & ... & (a_n)^{k-1} \end{pmatrix}$$

Cela s'explique facilement : le vecteur (1,0,0,0...,0) aura $P_{\vec{u}}(X) = 1$ donc une ligne remplie de 1. Le vecteur (0,1,0,0...,0) aura $P_{\vec{u}}(X) = X$ donc une ligne remplie de $a_1, a_2, a_3, ...$ La suite continue : la $3^{\text{ème}}$ ligne sera (0,0,1,0,0,...,0) qui donnera $P_{\vec{u}}(X) = X^2$ donc une ligne remplie de $(a_i)^2$, etc.

La forme systématique, la matrice de contrôle et le syndrome se calculent comme précédemment.

14.4 Notes et précisions sur les codes de Reed Solomon

- En choisissant les éléments $a_0, a_1, ...$ dans un autre ordre que le naturel, le code sera toujours juste et le "même", simplement que les mots seront dans un autre ordre.
- Pour corriger un mot qui a eu des effacements (pour autant que le poids de l'effacement soit acceptable), il suffit de chercher le syndrome du mot. Pour que le mot soit valide, le syndrome doit valoir 0. Cela nous donnera des équations, avec les effacements comme inconnues. Une simple résolution d'équations (avec des matrices par exemple) permet de trouver les effacements.
- Pour corriger les erreurs, c'est plus délicat à la main. Il faut connaître le poids de l'effacement. Chercher le syndrome du mot peut nous permettre de corriger avec un peu de jugeote. La série 12,

exercice 6 donne une bonne méthode. Sinon allez exhaustivement : chercher toutes les corrections possibles et tester les syndromes.