

Théorie des codes

Yves Denneulin, Jean-Louis Roch, Eric Tannier

20 janvier 2000

Table des matières

Introduction	5
1 La notion de code	7
1.1 Codes et systèmes de codage	7
1.1.1 Définitions	7
1.1.2 Déchiffabilité d'un code	8
1.1.3 Propriété du préfixe	10
1.2 L'arbre de Huffman	10
1.2.1 Définitions	10
1.2.2 Représentation des codes instantannés	11
1.3 Théorème de McMillan	11
2 Théorie de l'information	15
2.1 Source d'information	15
2.1.1 Source sans mémoire	15
2.1.2 Longueur moyenne d'un code	16
2.1.3 Extension d'une source	16
2.1.4 Source générale (avec mémoire)	16
2.2 Algorithme de codage de Huffman	16
2.2.1 Description de l'algorithme	17
2.2.2 L'algorithme de Huffman est optimal	18
2.3 Entropie d'une source	19
2.3.1 Quantité d'information	19
2.3.2 Entropie	20
2.4 Théorème de Shannon	21
2.5 Codes compresseurs usuels	23
3 Détection et correction d'erreurs	27
3.1 Formalisation du problème et définitions	28
3.1.1 Code systématique par blocs	28
3.1.2 Code correcteur et distance de Hamming	30
3.1.3 Code parfait	33
3.1.4 Cas particulier où $V = \{0, 1\}$ - Codes de Hamming	34

3.1.5	Codes cycliques : CRC	36
3.2	Bases mathématiques des codes cycliques	36
3.2.1	Corps fini	37
3.2.2	Anneau quotient $V[X]/P(X)$	37
3.2.3	Isomorphisme entre $V[X]/P$ et $V^{\deg(P)}$	38
3.2.4	PGCD et polynômes irréductibles	38
3.2.5	Factorisation de $X^n - 1$: classes cyclotomiques	40
3.3	Construction de codes cycliques	41
3.3.1	Codes linéaires	41
3.3.2	Codes cycliques	43
3.3.3	Construction d'un code cyclique : polynôme génerateur	44
3.3.4	Codage et décodage d'un code cyclique	45
3.3.5	Classes cyclotomiques et distance minimale	46
3.3.6	Codes BCH	47
3.4	Codes cycliques usuels	47
3.4.1	Codes de Reed-Solomon	48
3.4.2	Codes C.I.R.C.	49
3.4.3	Quelques autres codes cycliques	49
4	Méthodes de chiffrement	51
4.1	Introduction	51
4.1.1	Méthode d'attaque	52
4.2	Systèmes conventionnels à clé privée	53
4.2.1	Le DES	53
4.2.2	L'exponentiation	54
4.3	Cryptographie à clé publique	56
4.3.1	Secret et authentification	57
4.3.2	L'algorithme RSA	58
4.3.3	Utilisation de RSA	60

Introduction

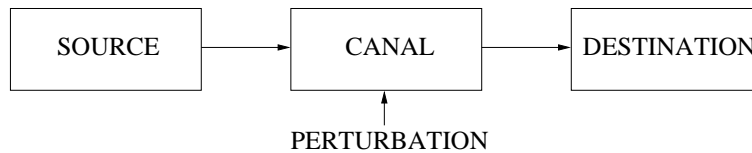


FIG. 1 – Schéma fondamental du codage

La transmission de messages entre un émetteur et un destinataire via un canal de communication introduit différents problèmes.

- Le canal permet de transmettre des signaux (typiquement des bauds). Il s'agit alors de transformer le message à émettre (par exemple un texte en français) en une séquence de signaux : on parle de *codage*. Le destinataire doit être capable de décoder la séquence de signaux reçus pour pouvoir lire le message émis.
- Dans un souci d'efficacité, la séquence de signaux doit être la plus courte possible. On s'intéresse donc à des codages qui minimisent la taille de la séquence émise : on parle de *compression*.
- Le medium peut introduire des erreurs : certains signaux émis peuvent être perdus ou altérés lors de la transmission. Dans ce cas, pour que l'émetteur puisse décoder correctement le message reçu, on utilise des codages spécifiques, permettant de *détection*, voire *corriger* les erreurs introduites lors de la transmission. On parle de *code détecteur/correcteur*.
- Le canal est généralement partagé par plusieurs couples émetteur-destinataire. Pour que l'émetteur puisse envoyer un message qui ne puisse être lu que par un destinataire spécifique (bien que la séquence de signaux associée soit visible par d'autres individus), le message doit être *crypté* grâce à un codage spécifique. La *cryptographie* étudie des codages et des protocoles permettant de communiquer des messages de manière secrète, de signer des documents ou d'authentifier un émetteur.

La théorie des codes et ses trois domaines d'application que sont la compression, la détection/correction d'erreurs et la cryptographie utilisent la

théorie des corps finis (dont on donnera un rappel des grands principes), et quelques éléments de probabilités, de théorie des graphes et de manipulation des chaînes de symboles.

Le premier chapitre de ce cours est une introduction à la notion de code, comprend les définitions et propriétés fondamentales concernant les objets mathématiques introduits. Les trois chapitres suivants, consacrés respectivement à la théorie de l'information, du codage détecteur/correcteur et de la cryptographie, présentent les résultats mathématiques fondamentaux et les algorithmes génériques qui en découlent. Chacun de ces trois chapitres est illustré par une utilisation pratique dans le contexte des télécommunications.

Chapitre 1

La notion de code

Ce chapitre donne les définitions de base et quelques propriétés utiles à la manipulation des codes.

1.1 Codes et systèmes de codage

1.1.1 Définitions

Un *vocabulaire* V est un ensemble fini de k éléments $\{v_1, \dots, v_k\}$. Le cardinal k de V est noté $|V|$.

Un *code* C sur un vocabulaire V est un sous-ensemble fini de V^+ (ensemble des chaînes sur V de longueur non nulle). Un élément c_i de C est appelé *mot de code*. Sa longueur est notée $l(c_i)$. L'*arité* du code est le cardinal de V .

Exemple : $C = \{0, 10, 110\}$ est un code d'arité 2 (on dit aussi binaire) sur le vocabulaire $V = \{0, 1\}$.

Dans tout ce cours, dans un souci de simplicité et de par leur importance pratique en télécommunications, on s'intéresse plus particulièrement aux codes binaires. Cependant, la plupart des résultats sont généralisables à des codes d'arité quelconque.

Soit $S = \{s_1, \dots, s_q\}$ un ensemble fini, qu'on appelle le *vocabulaire source*. On appelle un *message* une chaîne sur S . Soit C un code sur un vocabulaire V ; une *fonction d'encodage* de S dans C est une bijection $f : S \rightarrow C$. (C, f) est alors appelé *schéma de codage* pour S .

Exemple : Le sous-ensemble du codage ASCII pour les lettres majuscules est le suivant :

A	01000001	J	01001010	S	01010011
B	01000010	K	01001011	T	01010100
C	01000011	L	01001100	U	01010101
D	01000100	M	01001101	V	01010110
E	01000101	N	01001110	W	01010111
F	01000110	O	01001111	X	01011000
G	01000111	P	01010000	Y	01011001
H	01001000	Q	01010001	Z	01011010
I	01001001	R	01010010	espace	00100000

Pour communiquer un message $a_1 \dots a_n$ via un canal permettant de transmettre les mots d'un code C , on le traduit sous la forme $c_1 \dots c_n = f(a_1) \dots f(a_n)$. Le fait que f soit bijective ne suffit cependant pas pour que le message puisse être décodé sans ambiguïté par le récepteur.

Prenons l'exemple du codage des lettres de l'alphabet $S = \{A, \dots, Z\}$ par les entiers $C = \{0, \dots, 25\}$ écrits en base 10 :

$$f(A) = 0, f(B) = 1, \dots, f(J) = 9, f(K) = 10, f(L) = 11, \dots, f(Z) = 25.$$

Le codage 1209 peut alors correspondre à différents messages : par exemple, BUJ ou MAJ ou BCAJ.

Il est donc nécessaire d'ajouter des contraintes sur le code pour qu'un message quelconque puisse être déchiffré sans ambiguïté.

1.1.2 Déchiffrabilité d'un code

Un code C sur un vocabulaire V est dit *uniquement déchiffirable* (on dit parfois *non ambigu*) si et seulement si, pour tout $x = x_1 \dots x_n \in V^+$, il existe au plus une séquence $c = c_1 \dots c_m \in C^+$ telle que

$$c_1 \dots c_m = x_1 \dots x_n$$

Propriété 1 *Un code C sur un vocabulaire V est uniquement déchiffirable si et seulement si pour toutes séquences $c = c_1 \dots c_n$ et $d = d_1 \dots d_m$ de C^+ :*

$$c = d \implies (n = m \text{ et } \forall 1 \leq i \leq n, c_i = d_i)$$

Exemple :

- $C = \{0, 01, 001\}$ n'est pas uniquement déchiffirable.
- $C = \{01, 10\}$ est uniquement déchiffirable.
- $C = \{0, 10, 110\}$ est uniquement déchiffirable.

Théorème 1 (Kraft) *Il existe un code uniquement déchiffirable sur un vocabulaire V dont les mots $\{c_1, \dots, c_n\}$ sont de longueur l_1, \dots, l_n si et seulement si*

$$\sum_{i=1}^n \frac{1}{|V|^{l_i}} \leq 1$$

Preuve 1/ (\Rightarrow)

Soit C un code uniquement déchiffable, d'arité q .

Pour $1 \leq k \leq m = \max(\text{longueurs des mots de } C)$, Soit r_k le nombre de mots de longueur k .

Nous développons l'expression suivante, pour un entier quelconque u :

$$\left(\sum_{i=1}^n \frac{1}{q^{l_i}}\right)^u = \left(\sum_{k=1}^m \frac{r_k}{q^k}\right)^u$$

Chaque terme est de la forme

$$\frac{r_{i_1} \dots r_{i_u}}{q^{i_1 + \dots + i_u}}$$

Et en regroupant pour chaque valeur $s = i_1 + \dots + i_u$, on obtient les termes

$$\sum_{i_1 + \dots + i_u = s} \frac{r_{i_1} \dots r_{i_u}}{q^s}$$

Soit $N(s) = \sum_{i_1 + \dots + i_u = s} r_{i_1} \dots r_{i_u}$. L'expression initiale s'écrit :

$$\left(\sum_{i=1}^n \frac{1}{q^{l_i}}\right)^u = \sum_{s=u}^{mu} \frac{N(s)}{q^s}$$

$N(s)$ est le nombre de combinaisons de mots de C de longueur totale s . Comme C est uniquement déchiffable, deux combinaisons ne peuvent être égales au même mot sur le vocabulaire de C , de taille q , $N(s)$ est inférieur au nombre total de messages de longueur s sur ce vocabulaire, soit q^s . Donc

$$\left(\sum_{i=1}^n \frac{1}{q^{l_i}}\right)^u \leq mu - m + 1 \leq mu$$

Et en prenant la u -ième racine,

$$\sum_{i=1}^n \frac{1}{q^{l_i}} \leq (mu)^{1/u}$$

Si u tend vers l'infini,

$$\sum_{i=1}^n \frac{1}{q^{l_i}} \leq 1$$

2/ (\Leftarrow)

Ce résultat est une conséquence du théorème de McMillan, que nous verrons plus loin dans ce chapitre. \square

1.1.3 Propriété du préfixe

On dit qu'un code C sur un vocabulaire V a la *propriété du préfixe* (on dit parfois qu'il est *instantané*, ou *irréductible*) si et seulement si pour tout couple de mots de code distincts (c_1, c_2) , c_2 n'est pas un préfixe de c_1 .

Exemple : $a = 101000$, $b = 01$, $c = 1010$: b n'est pas un préfixe de a mais c est un préfixe de a .

Grâce à la propriété du préfixe, on peut déchiffrer les mots d'un tel code dès la fin de la réception du mot (instantanéité), ce qui n'est pas toujours le cas pour les codes uniquement déchiffrables : par exemple, si $V = 0, 01, 11$, et si on reçoit le message $m = 001111111111\dots$, il faut attendre l'occurrence suivante d'un 0 pour pouvoir déchiffrer le second mot (0 ou 01?).

Propriété 2 *Tout code possédant la propriété du préfixe est uniquement déchiffrable.*

Preuve Soit un code C sur V qui n'est pas uniquement déchiffrable. Alors il existe une chaîne $a \in V^n$ telle que $a = c_1 \dots c_l = d_1 \dots d_k$, les c_i et d_i étant des mots de C et $c_i \neq d_i$ pour au moins un i . Choisissons le plus petit i tel que $c_i \neq d_i$ ($\forall j < i, c_j = d_j$). Alors $l(c_i) \neq l(d_i)$. Si $l(c_i) < l(d_i)$, c_i est un préfixe de d_i et dans le cas contraire d_i est un préfixe de c_i . C n'a donc pas la propriété du préfixe. \square

La réciproque est fautive : le code $C = \{0, 01\}$ est uniquement déchiffrable, mais ne possède pas la propriété du préfixe.

Propriété 3 *Tout code dont tous les mots sont de même longueur possède la propriété du préfixe.*

1.2 L'arbre de Huffman

1.2.1 Définitions

On donne ici les définitions dans le cas binaire, mais elles sont généralisables pour des codes d'arité quelconque.

On appelle un *arbre de Huffman* un arbre binaire tel que tout sous-arbre a soit 0 soit 2 fils (il est localement complet). Dans ce dernier cas, on assigne le symbole "1" à l'arête reliant la racine locale au fils gauche et "0" au fils droit.

A chaque feuille d'un arbre de Huffman, on peut associer un mot de $\{0, 1\}^+$: C'est la chaîne des symboles marquant les arêtes d'un chemin depuis la racine jusqu'à la feuille. On appelle *code de Huffman* l'ensemble des mots correspondant aux chemins d'un arbre de Huffman.

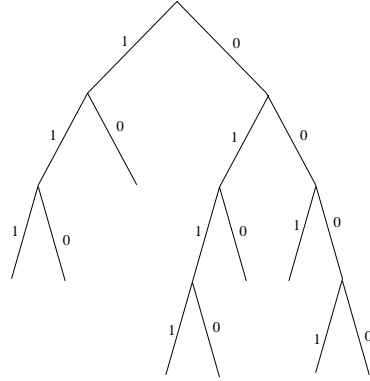


FIG. 1.1 – Exemple d’arbre de Huffman

Exemple : le code correspondant à l’arbre de la figure 1.1 est :

$$\{111, 110, 10, 0111, 0110, 010, 001, 0001, 0000\}$$

1.2.2 Représentation des codes instantannés

Propriété 4 *Un code de Huffman possède la propriété du préfixe.*

Preuve Si un mot de code c_1 est un préfixe de c_2 , alors le chemin représentant c_1 dans l’arbre de Huffman est inclus dans le chemin représentant c_2 . Comme c_1 et c_2 sont, par définition, associés à des feuilles de l’arbre, $c_1 = c_2$. Il n’existe donc pas deux mots différents dont l’un est le préfixe de l’autre, et le code de Huffman a la propriété du préfixe. \square

Propriété 5 *Tout code qui possède la propriété du préfixe est contenu dans un code de Huffman.*

Preuve On construit un arbre de Huffman à partir d’un code C quelconque possédant la propriété du préfixe :

Soit un arbre de Huffman complet (toutes les feuilles sont à distance constante de la racine) de hauteur $l = \max(\text{longueurs des mots de } C)$. Chaque mot c_i de C est associé à un chemin depuis la racine jusqu’à un noeud. On peut alors élaguer le sous-arbre dont ce noeud est racine (tous les mots pouvant être représentés dans les noeuds de ce sous-arbre ont c_i pour préfixe). Tous les mots de C sont toujours dans les noeuds de l’arbre résultant. On peut effectuer la même opération pour tous les mots. On a finalement un code de Huffman contenant tous les mots de C . \square

1.3 Théorème de McMillan

Théorème 2 (McMillan) *Sur un vocabulaire V , il existe un code qui possède la propriété du préfixe dont les mots $\{c_1, \dots, c_n\}$ sont de longueur*

l_1, \dots, l_n si et seulement si

$$\sum_{i=1}^n \frac{1}{|V|^{l_i}} \leq 1$$

Preuve 1/ preuve de (\Rightarrow) : on suppose qu'il existe un code instantané avec les longueurs de mots l_1, \dots, l_n . On a montré qu'on peut construire sa représentation par un arbre de Huffman. L'arbre complet initial a pour hauteur l , et $|V|^l$ feuilles. On compte à chaque opération d'élagage le nombre de feuilles de l'arbre initial complet qu'on enlève ainsi à l'arbre de Huffman. Pour un mot de longueur l_i , c'est le nombre de feuilles de l'arbre complet de hauteur $l - l_i$, soit $|V|^{l-l_i}$ (on suppose enlevée, car non réutilisée, une feuille d'un sous-arbre de hauteur 0). Pour toutes les opérations, on enlève $\sum_{i=1}^n |V|^{l-l_i}$ feuilles. Mais on ne peut en enlever au total plus que le nombre initial, soit

$$\sum_{i=1}^n |V|^{l-l_i} \leq |V|^l$$

d'où

$$\sum_{i=1}^n \frac{1}{|V|^{l_i}} \leq 1$$

2/ preuve de (\Leftarrow) : on sait que l'inégalité est satisfaite. On cherche à construire un arbre de Huffman dont les mots de code ont pour longueur l_1, \dots, l_n , mots que l'on suppose classés par ordre croissant de leurs longueurs. Pour pouvoir choisir un noeud dans l'arbre complet pour un mot de longueur l_k , il doit y avoir dans l'arbre au moins $|V|^{l-l_k}$ feuilles de l'arbre complet initial. Si on a déjà "placé" les mots de longueur l_1, \dots, l_{k-1} , on a enlevé $\sum_{i=1}^{k-1} |V|^{l-l_i}$ feuilles de l'arbre initial par les successives opérations d'élagage. Il reste donc $|V|^l(1 - \sum_{i=1}^{k-1} |V|^{-l_i})$ feuilles. Or, on sait d'après l'inégalité de Kraft que

$$\sum_{i=k}^n \frac{1}{|V|^{l_i}} \leq 1 - \sum_{i=1}^{k-1} \frac{1}{|V|^{l_i}}$$

soit

$$|V|^l(1 - \sum_{i=1}^{k-1} |V|^{-l_i}) \geq \sum_{i=k}^n |V|^{l-l_i} \geq |V|^{l-l_k}$$

On peut donc placer le mot de longueur l_k , et en réitérant l'opération construire l'arbre de Huffman. Le code de Huffman, de longueurs de mots l_1, \dots, l_n , vérifie la propriété du préfixe.

Remarque : Ceci prouve l'implication laissée en suspens du théorème de Kraft. \square

Corollaire 1 *S'il existe un code uniquement déchiffrable dont les mots sont de longueur l_1, \dots, l_n , alors il existe un code instantané de mêmes longueurs de mots.*

C'est une conséquence des théorèmes de Kraft et McMillan. Les codes déchiffrables qui ne possèdent pas la propriété du préfixe ne produisent pas de code aux mots plus courts que les codes instantanés, auxquels on peut donc se restreindre pour la compression d'information (leurs propriétés les rendent plus maniables).

Chapitre 2

Théorie de l'information

Un premier intérêt de la théorie mathématique des codes réside dans l'optimisation possible de la taille d'un message qui doit transiter par un canal, ou être stocké sur un support. On effectue donc ce codage à la sortie de la source, et on décode à l'entrée de la destination. Nous supposons ici que le canal n'est pas soumis aux perturbations (on parle de codage sans bruit), le codage de canal et la gestion des erreurs sont étudiés au chapitre suivant. Il existe des techniques d'encodage permettant de choisir des codes efficaces ainsi qu'une théorie importante, originairement développée par Shannon en 1948 permettant de quantifier l'information contenue dans un message et de calculer la taille minimale d'un schéma de codage, et de connaître ainsi la "valeur" d'un code donné.

2.1 Source d'information

2.1.1 Source sans mémoire

Définition 1 *On définit une source d'information par un couple $\mathcal{S} = (S, \mathcal{P})$ où $S = (s_1, \dots, s_n)$ est un vocabulaire source et $\mathcal{P} = (p_1, \dots, p_n)$ est une distribution de probabilité. $\forall i, p_i$ est la probabilité d'occurrence de s_i dans S . Pour une telle source, dite sans mémoire, les événements (occurrences d'un symbole) sont considérés comme indépendants, et leur probabilité reste stable au cours de l'émission (la source est stationnaire).*

Définition 2 *Une source \mathcal{S} est dite sans redondance si tous les symboles de S apparaissent avec la même probabilité ($p_1 = \dots = p_n = \frac{1}{n}$).*

Nous allons montrer dans ce chapitre qu'une source redondante peut être codée de manière à éliminer la redondance dans l'vocabulaire lisible par le canal. Nous précisons donc le schéma de codage défini en introduction de ce cours :

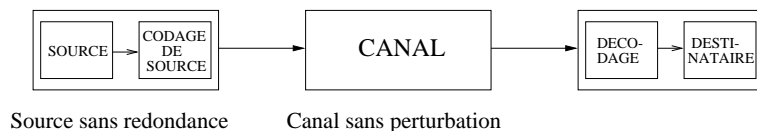


FIG. 2.1 – Codage de source

2.1.2 Longueur moyenne d'un code

Définition 3 Soient $\mathcal{S} = (S, \mathcal{P})$ une source, (C, f) un schéma d'encodage de S . La longueur moyenne de (C, f) est :

$$l(C, f) = \sum_{i=1}^n l(f(s_i))P(s_i)$$

Exemple : $S = \{a, b, c, d\}$, $\mathcal{P} = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})$, $V = \{0, 1\}$.

Si $C = \{f(a) = 00, f(b) = 01, f(c) = 10, f(d) = 11\}$, la longueur moyenne du schéma est 2.

Si $C = \{f(a) = 0, f(b) = 10, f(c) = 110, f(d) = 1110\}$, la longueur moyenne du schéma est : $1 * \frac{1}{2} + 2 * \frac{1}{4} + 3 * \frac{1}{8} + 4 * \frac{1}{8} = 1.875$

Nous utilisons la longueur moyenne d'un schéma d'encodage pour mesurer son efficacité.

2.1.3 Extension d'une source

Définition 4 Soit une source \mathcal{S} sans mémoire. La k -ième extension \mathcal{S}^k de \mathcal{S} est le doublet (S^k, \mathcal{P}^k) , où S^k est l'ensemble des mots de longueur k sur S , et \mathcal{P}^k est la distribution de probabilité ainsi définie : pour un mot $s = s_{i_1} \dots s_{i_k} \in S^k$, $P^k(s) = P(s_{i_1} \dots s_{i_k}) = P(s_{i_1}) \dots P(s_{i_k})$.

Exemple : $S = (s_1, s_2)$, $\mathcal{P} = (\frac{1}{4}, \frac{3}{4})$

alors $S^2 = (s_1s_1, s_1s_2, s_2s_1, s_2s_2)$ et $P^2 = (\frac{1}{16}, \frac{3}{16}, \frac{3}{16}, \frac{9}{16})$

2.1.4 Source générale (avec mémoire)

À COMPLÉTER

2.2 Algorithme de codage de Huffman

Cette méthode permet de trouver le meilleur schéma d'encodage d'une source sans mémoire \mathcal{S} .

2.2.1 Description de l'algorithme

La source à coder est $\mathcal{S} = (S, \mathcal{P})$, le vocabulaire de codage V . Il est nécessaire à l'optimalité du résultat de vérifier que $|V| - 1$ divise $|\mathcal{S}| - 1$ (afin d'obtenir un arbre localement complet). Dans le cas contraire, il est facile de rajouter des symboles à S , de probabilités d'occurrence nulle, jusqu'à ce que $|V| - 1$ divise $|\mathcal{S}| - 1$. Les mots de codes associés (les plus longs) ne seront pas utilisés.

On construit avec le vocabulaire source S un ensemble de noeuds isolés auxquels on associe les probabilités de P .

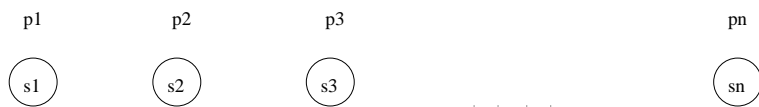


FIG. 2.2 – algorithme de Huffman : départ

Soient $p_{i_1}, \dots, p_{i_{|V|}}$ les $|V|$ symboles de plus faibles probabilités. On construit un arbre (sur le modèle des arbres de Huffman), dont la racine est un nouveau noeud et auquel on associe la probabilité $p_{i_1} + \dots + p_{i_{|V|}}$, et dont les branches sont incidentes aux noeuds $p_{i_1}, \dots, p_{i_{|V|}}$. La figure 2.3 montre un exemple de cette opération pour $|V| = 2$.

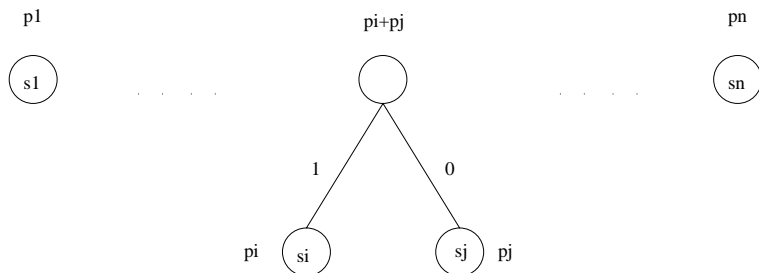


FIG. 2.3 – algorithme de Huffman : première étape ($|V| = 2$)

On recommence ensuite avec les $|V|$ plus petites valeurs parmi les noeuds du plus haut niveau (les racines), jusqu'à n'obtenir qu'un arbre (à chaque itération, il y a $|V| - 1$ éléments en moins parmi les noeuds de plus haut niveau), dont les mots de S sont les feuilles, et dont les mots de code associés dans le schéma ainsi construit sont les mots correspondant aux chemins de la racine aux feuilles.

Exemple : Soit la source à coder sur $V = \{0, 1\}$

Symbole	Probabilité
a	0,35
b	0,10
c	0,19
d	0,25
e	0,06
f	0,05

Les étapes successives de l'algorithme sont décrites par la figure 2.4.
Le code de Huffman construit est alors :

Symbole	Mot de code
a	11
b	010
c	00
d	10
e	0111
f	0110

2.2.2 L'algorithme de Huffman est optimal

Théorème 3 *Le code issu de l'algorithme de Huffman est optimal parmi tous les codes de S sur V .*

Preuve À COMPLÉTER \square

On peut pourtant obtenir des codes plus efficaces à partir des extensions de la source, comme on peut le voir à partir d'un exemple :

Soit $\mathcal{S} = (S, \mathcal{P})$, $S = (s_1, s_2)$, $\mathcal{P} = (1/4, 3/4)$. Un codage de Huffman pour \mathcal{S} donne évidemment $s_1 \rightarrow 0$ et $s_2 \rightarrow 1$, et sa longueur moyenne est 1.

Un codage de Huffman pour $\mathcal{S}^2 = (S^2, \mathcal{P}^2)$ donne :

$$\begin{aligned} s_1 s_1 &\rightarrow 010 \\ s_1 s_2 &\rightarrow 011 \\ s_2 s_1 &\rightarrow 00 \\ s_2 s_2 &\rightarrow 1 \end{aligned}$$

et sa longueur moyenne est $3 * \frac{1}{16} + 3 * \frac{3}{16} + 2 * \frac{3}{16} + \frac{9}{16} = \frac{27}{16} = 1,6875$

La longueur moyenne de ce code est donc $l = 1,6875$, et en comparaison avec le code sur \mathcal{S} (les mots de \mathcal{S}^2 sont de longueur 2), $l = 1,6875/2 = 0,84375$, ce qui est meilleur que le code sur la source originelle.

Nous pouvons encore améliorer ce codage en examinant la source \mathcal{S}^3 . Il est aussi possible d'affiner le codage par une meilleure modélisation de la source : souvent, l'occurrence d'un symbole n'est pas indépendante des symboles précédemment émis par une source (dans le cas d'un texte, par exemple). Dans ce cas, les probabilités d'occurrence sont conditionnelles et

il existe des modèles (le modèle de Markov, en particulier) qui permettent un meilleur codage. Mais ces procédés ne conduisent pas à des améliorations infinies. Il existe un seuil pour la longueur moyenne, appelé entropie, ne dépendant que de la source, en deçà duquel on ne peut pas trouver de code. Ce résultat théorique important fait l'objet des sections suivantes.

2.3 Entropie d'une source

2.3.1 Quantité d'information

Nous arrivons aux notions fondamentales de la théorie de l'information. Il est important, au-delà des définitions mathématiques de la quantité d'information et de l'entropie (que seul le théorème de Shannon pourrait justifier), de saisir la signification de ces grandeurs, et c'est pourquoi nous commençons par cette approche intuitive.

Soit une source $\mathcal{S} = (S, \mathcal{P})$. Nous ne connaissons de cette source qu'une distribution de probabilité, mais nous cherchons à mesurer quantitativement à quel point nous ignorons le comportement de \mathcal{S} . Par exemple, cette *incertitude* est plus grande si le nombre de symboles dans S est plus grand. Elle est faible si une probabilité p_i est proche de 1, et plus forte en cas d'équiprobabilité.

Afin de choisir une fonction qui quantifie l'incertitude, nous cherchons une mesure de l'information I contenue dans un événement x (pour une source, c'est l'occurrence d'un symbole s apparaissant avec une probabilité p). C'est une fonction croissante de l'*improbabilité* de cet événement.

Par exemple, imaginons une source pouvant émettre deux symboles, l'un avec une probabilité $p_1 = \frac{1}{100}$, l'autre avec une probabilité $p_2 = \frac{99}{100}$. L'apparition du premier symbole fournira au destinataire plus d'information sur la source (il ne lui manque que le second, qui apparaîtra avec une forte probabilité) que l'apparition du second, qui laissera le destinataire pratiquement aussi peu informé sur la source qu'avant l'apparition du symbole.

Nous devons donc choisir $I(x)$ fonction inverse de la probabilité d'occurrence de x . $I(x) = f(\frac{1}{P(x)})$. Soit $I(p) = f(\frac{1}{p})$. D'autres axiomes sont imposés par cette approche intuitive :

- la fonction quantité d'information $I(p)$ est positive et continue.
- l'information apportée par un événement certain est nulle.
- pour deux événements indépendants x, y , la quantité d'information apportée par x et y sera la somme des quantités d'information $I(x)$ et $I(y)$; puisque la probabilité d'occurrence de deux événements est le produit des probabilités, et que la quantité d'information I est fonction inverse de la probabilité de x , I vérifie $I(x, y) = I(x) + I(y)$, soit $f(\frac{1}{P(x)P(y)}) = f(\frac{1}{P(x)}) + f(\frac{1}{P(y)})$.

Les seules fonctions satisfaisant ces axiomes sont :

$$I(x) = K * \log\left(\frac{1}{P(x)}\right)$$

ce qui est un résultat classique que nous ne redémontrons pas. Choisir une constante K revient à choisir une base pour le logarithme, ce qui est équivalent à choisir une unité de mesure. Par convention, on choisit le \log_2 et l'unité *bit* pour "binary unit", ou parfois *shannon*. Nous arrivons à la définition :

Définition 5 Pour une source $\mathcal{S} = (S, \mathcal{P})$, La quantité d'information fournie par l'occurrence d'un symbole $s \in S$ de probabilité $p \in]0, 1]$ est :

$$I(p) = \log_2\left(\frac{1}{p}\right)$$

2.3.2 Entropie

L'entropie d'une source est la quantité moyenne d'information contenue dans cette source.

Définition 6 L'entropie d'une source $\mathcal{S} = (S, \mathcal{P})$, $S = (s_1, \dots, s_n)$, $\mathcal{P} = (p_1, \dots, p_n)$ est :

$$H(\mathcal{S}) = H(p_1, \dots, p_n) = \sum_{i=1}^n p_i \log_2\left(\frac{1}{p_i}\right)$$

C'est une mesure de *l'incertitude* liée à une loi de probabilités, ce qui est illustré par l'exemple suivant : On considère la variable aléatoire (source) issue du jet d'un dé à n faces. Il y a plus d'incertitude dans le résultat de ce jet si le dé est normal que si le dé est biaisé. Ce qui se traduit par $\forall p_1, \dots, p_n$, $H(p_1, \dots, p_n) \leq H\left(\frac{1}{n}, \dots, \frac{1}{n}\right) = \log_2 n$

En généralisant,

Propriété 6 Soit $\mathcal{S} = (S, \mathcal{P})$ une source, de loi (p_1, \dots, p_n) .

$$H(\mathcal{S}) \leq \log_2 n$$

Preuve Nous commençons par montrer un lemme dont nous aurons plusieurs fois besoin.

lemme de Gibbs : Soient (p_1, \dots, p_n) , (q_1, \dots, q_n) deux lois de probabilité discrètes.

$$\sum_{i=1}^n p_i * \log \frac{q_i}{p_i} \leq 0$$

Preuve du lemme : on sait que $\forall x \in \mathbb{R}$, $\ln(x) \leq x - 1$. Donc

$$\sum_{i=1}^n p_i * \ln \frac{q_i}{p_i} \leq \sum_{i=1}^n p_i * \left(\frac{q_i}{p_i} - 1\right)$$

Soit puisque $\sum_{i=1}^n p_i = \sum_{i=1}^n q_i = 0$,

$$\sum_{i=1}^n p_i * \ln \frac{q_i}{p_i} \leq 0$$

Par suite,

$$\sum_{i=1}^n p_i * \log \frac{q_i}{p_i} \leq 0$$

Appliquons ensuite ce lemme à la distribution $(q_1, \dots, q_n) = (\frac{1}{n}, \dots, \frac{1}{n})$, on obtient

$$H(\mathcal{S}) \leq \log_2 n$$

□

Propriété 7 Soient \mathcal{S} une source, et \mathcal{S}^k sa k -ième extension.

$$H(\mathcal{S}^k) = kH(\mathcal{S})$$

Preuve On développe l'expression de $H(\mathcal{S}^k)$:

$$H(\mathcal{S}^k) = \sum_{i_1, \dots, i_k, 0 \leq i_j \leq |V|} p_{i_1} \dots p_{i_k} \log \frac{1}{p_{i_1} \dots p_{i_k}}$$

Soit

$$H(\mathcal{S}^k) = \sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} \log \frac{1}{p_{i_1}} + \dots + \sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} \log \frac{1}{p_{i_k}}$$

Le premier terme de cette somme est :

$$\sum_{i_1, \dots, i_k} p_{i_1} \dots p_{i_k} \log \frac{1}{p_{i_1}} = \sum_{i_1=1}^{|V|} p_{i_1} \log \frac{1}{p_{i_1}} * \sum_{i_2=1}^{|V|} p_{i_2} * \dots * \sum_{i_k=1}^{|V|} p_{i_k}$$

Puisque la somme des probabilités p_j vaut toujours 1, ceci est égal à :

$$\sum_{i_1=1}^{|V|} p_{i_1} \log \frac{1}{p_{i_1}} = H(\mathcal{S})$$

Et en remplaçant ce terme dans l'expression précédente,

$$H(\mathcal{S}^k) = H(\mathcal{S}) + \dots + H(\mathcal{S}) = kH(\mathcal{S})$$

□

2.4 Théorème de Shannon

Ce théorème fondamental de la théorie de l'information est connu sous le nom de *théorème de Shannon* ou *théorème du codage sans bruit*

Nous commençons par énoncer le théorème dans le cas d'une source sans mémoire :

Théorème 4 Soit une source \mathcal{S} sans mémoire d'entropie H .

Tout code uniquement déchiffrable de \mathcal{S} sur un vocabulaire de taille Q , de longueur moyenne l , vérifie :

$$l \geq \frac{H}{\log_2 Q}$$

De plus, il existe un code uniquement déchiffrable de \mathcal{S} sur un vocabulaire de taille Q , de longueur moyenne l , qui vérifie :

$$l < \frac{H}{\log_2 Q} + 1$$

Preuve

première partie : Soit $C = (c_1, \dots, c_n)$ un code de \mathcal{S} uniquement déchiffrable sur un vocabulaire de taille Q , et (l_1, \dots, l_n) les longueurs des mots de C . soit $K = \sum_{i=1}^n \frac{1}{Q^{l_i}}$, $K \leq 1$ d'après le théorème de Kraft. Soient (q_1, \dots, q_n) tels que $\forall i, q_i = \frac{Q^{-l_i}}{K}$. On a alors $\forall i, q_i \in [0, 1]$ et $\sum_{i=1}^n q_i = 1$ donc (q_1, \dots, q_n) est une distribution de probabilités. On peut donc appliquer le lemme de Gibbs (démontré en section 2.3.2) :

$$\sum_{i=1}^n p_i \log \frac{q_i}{p_i} \leq 0$$

Soit ici,

$$\sum_{i=1}^n p_i \log \frac{Q^{-l_i}}{K p_i} \leq 0$$

Ou encore

$$\sum_{i=1}^n p_i \log \frac{1}{p_i} \leq \sum_{i=1}^n p_i l_i \log Q + \log K$$

Et comme $\log K \leq 0$,

$$H(\mathcal{S}) \leq l * \log Q$$

D'où le résultat.

seconde partie : Soient $l_i = \lfloor \log_Q \frac{1}{p_i} \rfloor$. Comme $\sum_{i=1}^n \frac{1}{Q^{l_i}} \leq 1$ (car $Q^{l_i} \geq \frac{1}{p_i}$), il existe un code de \mathcal{S} sur un vocabulaire de taille Q , uniquement déchiffrable, avec des longueurs de mots égales à (l_1, \dots, l_n) . Comme $l_i < \lfloor \log_Q \frac{1}{p_i} \rfloor + 1$, $p_i < Q^{-l_i+1}$ et par suite,

$$\sum_{i=1}^n p_i \log \frac{1}{p_i} > \sum_{i=1}^n p_i l_i \log Q - \log Q$$

Soit

$$l < \frac{H(\mathcal{S})}{\log Q} + 1$$

□

On en déduit le théorème pour la k -ième extension de \mathcal{S} :

Théorème 5 *Soit une source \mathcal{S} stationnaire d'entropie H .*

Tout code uniquement déchiffrable de \mathcal{S}^k sur un vocabulaire de taille Q , de longueur moyenne l_k , vérifie :

$$\frac{l_k}{k} \geq \frac{H}{\log_2 Q}$$

De plus, il existe un code uniquement déchiffrable de \mathcal{S}^k sur un vocabulaire de taille Q , de longueur moyenne l_k , qui vérifie :

$$\frac{l_k}{k} < \frac{H}{\log_2 Q} + 1/k$$

Preuve la preuve de ce théorème est immédiate d'après la propriété selon laquelle $H(\mathcal{S}^k) = k * H(\mathcal{S})$. \square

Pour une source stationnaire quelconque, le théorème peut s'énoncer :

Théorème 6 *Pour toute source stationnaire d'entropie H , il existe un procédé de codage uniquement déchiffrable sur un vocabulaire de taille Q , et de longueur moyenne l , aussi proche que l'on veut de sa borne inférieure $H/\log_2(Q)$.*

En théorie, il est donc possible de trouver un code s'approchant indéfiniment de l'entropie. En pratique pourtant, si le procédé de codage consiste à coder les mots d'une extension de la source, on est limité évidemment par le nombre de ces mots ($|\mathcal{S}^k| = |\mathcal{S}|^k$, ce qui peut représenter un très grand nombre de mots).

L'algorithme de Huffman (ou des variantes) est utilisé en pratique, souvent en même temps que d'autres procédés de codages, pas toujours optimaux en théorie, mais qui font des hypothèses raisonnables sur la forme des fichiers à compresser (ce sont des modèles de source) pour diminuer l'entropie, ou accepter une destruction d'information que l'on suppose sans conséquence pour l'utilisation des donnés.

2.5 Codes compresseurs usuels

- L'algorithme de la commande "pack" de Unix est une implémentation de l'arbre de Huffman.
- L'algorithme de Lempel-Ziv (utilisé par les commandes "compress" de Unix, ou "gzip") est une variante sur le code de Huffman : deux arbres sont implémentés, un pour les chaînes de caractères, l'autre pour les distances entre occurrences. Les distances entre deux occurrences d'une chaîne sont bornées : lorsque la distance devient trop grande, on repart à 0.

- Il y a aussi des compressions avec perte d'information : par exemple, le codage au format JPEG (Joint Photographic Experts Group) compresse des images fixes. L'algorithme de codage est complexe, et se déroule en plusieurs étapes. Le principe de base est que les couleurs des pixels voisins dans une image diffèrent peu. On code donc le décalage par rapport à la pixel voisine. De plus, une image est un signal : plutôt que les valeurs des pixels, on calcule les fréquences (transformée de Fourier DFT, transformée en cosinus discrète DCT). En ne gardant que les premiers termes de la décomposition (les plus importants), on perd un peu d'information mais l'image reste visible. L'image est finalement codée comme une suite de nombres (une valeur de DCT suivie du nombre de pixels ayant cette valeur qui sont consécutives selon un balayage en zigzag de l'image). En fin d'algorithme, un codage de Huffman compresse le fichier de nombres ainsi obtenus.

- Le format MPEG (Motion Picture Experts Group) assure la compression d'images animées). L'algorithme de codage utilise JPEG pour coder une image mais prend en compte le fait que deux images consécutives dans une séquence vidéo sont très voisines. Une des particularités des normes MPEG est de chercher à compenser le mouvement (ex. zoom etc) d'une image à la suivante. Une sortie MPEG(-1) contient 4 sortes d'images : des images au format JPEG, des images codées par différences avec l'image précédente, des images bidirectionnelles codées par différence avec l'image précédente et la suivante, enfin des images basse résolution utilisées pour l'avance rapide sur un magnétoscope.

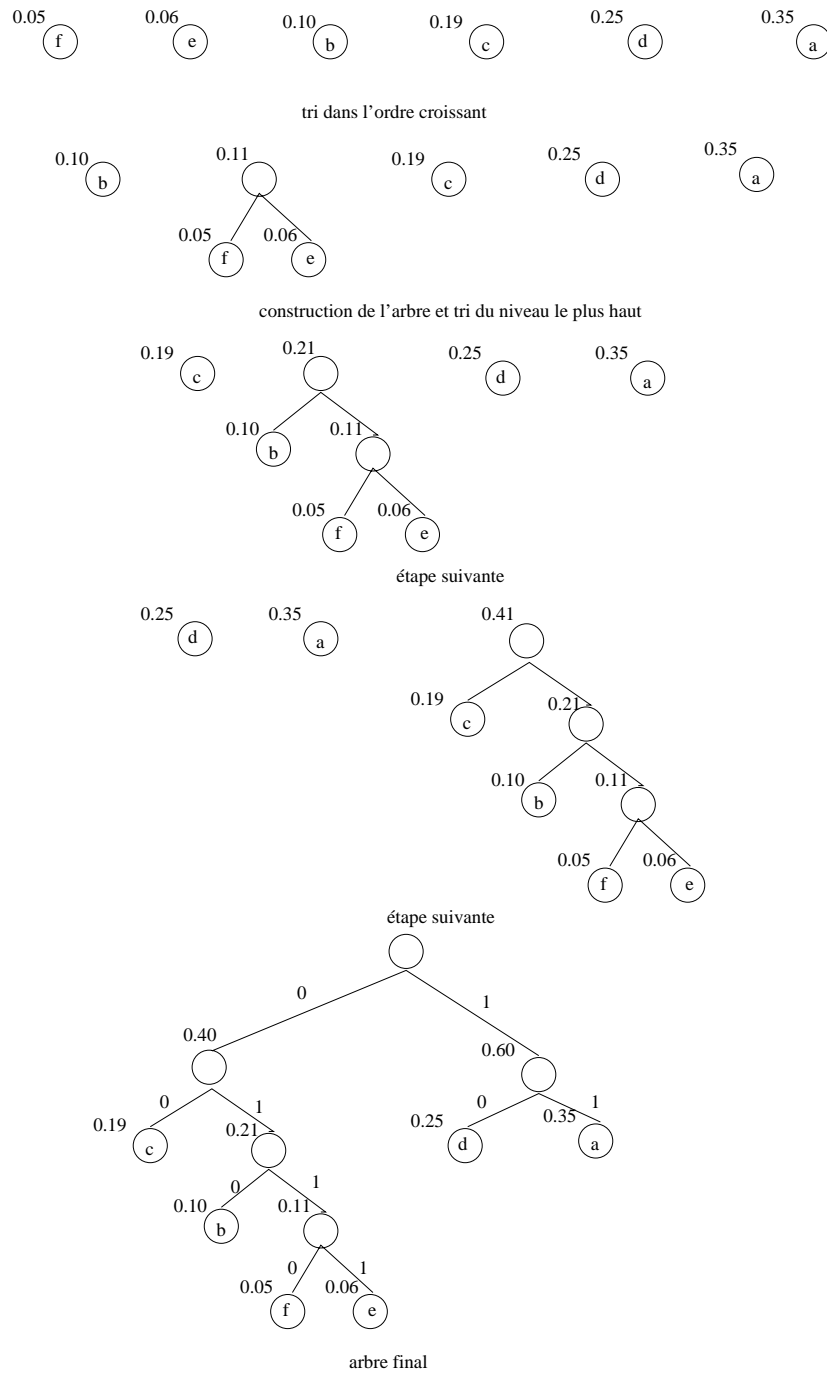


FIG. 2.4 – Exemple de construction d'un code de Huffman

Chapitre 3

Détection et correction d'erreurs

Lorsqu'une source émet une séquence de bits $S = s_1, \dots, s_k$, la séquence $S' = s'_1, \dots, s'_k$ reçue par le collecteur peut différer : si $s'_i \neq s_i$, on dit qu'il y a eu erreur sur le bit d'indice i .

Le taux d'erreur dépend de la nature de la ligne de transmission (locale/internationale, nombre de répéteurs, support câble/satellite, ...); il varie typiquement de 10^{-4} à 10^{-7} .

Les méthodes pour se protéger contre ces erreurs en les détectant voire en les corrigeant de manière automatique, consistent à ajouter des bits d'information supplémentaires, appelés *bits de contrôle*, à l'aide d'un mécanisme appelé *codeur*. Le codeur transforme donc la séquence à transmettre $S = s_1, \dots, s_k$ en une séquence $\phi(S) = s'_1, \dots, s'_k, \dots, s'_{k+r}$ qui comporte r bits de redondance par rapport au message initial.

Pour permettre le décodage, la fonction ϕ doit bien sûr être injective; ainsi, lorsqu'il n'y a pas d'erreur, le calcul après réception de $\phi^{-1}(s'_1, s'_{k+r}) = S$ permet de reconstruire le message source.

ϕ étant injective, $E_\phi = \text{Im}(\phi)$ est un sous-ensemble $E_\phi \subset \{0, 1\}^{k+r}$ comportant seulement 2^k mots de code (chacun codé sur $k+r$ bits). Ainsi, la réception d'un mot qui n'est pas dans E_ϕ (i.e. dans $\overline{E_\phi} = \{0, 1\}^{k+r} - E_\phi$) indique une erreur. La détection d'erreur repose alors sur le test de non-appartenance à E_ϕ . Comme il est possible de recevoir un mot comportant des erreurs mais appartenant à E_ϕ , on parle de *syndrome d'erreur*.

Dans un souci d'efficacité, le code ϕ doit être choisi pour que ce test :

- permette de détecter le plus d'erreurs possibles;
- soit le moins coûteux possible.

Suite à une détection d'erreurs, le décodeur peut éventuellement procéder à une correction, on distingue deux classes de corrections :

- correction directe : le signal erroné reçu contient suffisamment d'information pour permettre de retrouver le mot émis;

Mot de code sur 7 bits	Mot avec bit de parité
0101001	010100 1
0001001	000100 1 0
0000000	000000 0 0

FIG. 3.1 – Détection d'erreur par bit de parité.

- correction par retransmission (ou ARQ) : une demande de retransmission du message source est effectuée lorsque l'erreur ne peut pas être corrigée.

Le but de ce chapitre est d'introduire les principaux codes détecteurs/correcteurs d'erreurs, les codes *cycliques* (on dit aussi polynômiaux), et leurs fondements mathématiques.

Un exemple simple de détection : bit de parité longitudinale. Soit $S = s_1, \dots, s_k$; soit alors $\phi(S) = s_1, \dots, s_k, s_{k+1}$ où $s_{k+1} = (\sum_{i=1}^k s_i) \bmod 2$ indique¹ si S contient un nombre pair ou impair de 1.

De manière évidente (cf figure 3.1), l'ajout d'un bit de parité longitudinale permet de détecter toutes les erreurs portant sur un nombre impair de bits.

Un exemple simple de correction directe : contrôle de parité longitudinale et transversale. Corriger une erreur se ramène à localiser les bits où une erreur est apparue.

3.1 Formalisation du problème et définitions

3.1.1 Code systématique par blocs

Soit V le vocabulaire de base ; les éléments de V sont appelés des *chiffres*. On supposera de plus que les opérations d'addition (+) et de multiplication (\times) de chiffres dans V sont définies et confèrent à V une structure d'anneau commutatif :

- le chiffre "0" (resp. "1") est l'élément neutre pour + (resp \times) ;
- $(V, +, 0)$ est un groupe commutatif, i.e. + est associative et commutative et tout élément $x \in V$ a un inverse pour l'addition, noté $-x$.
- \times est associative et commutative (car anneau commutatif).
- \times est distributive sur + : $x \times (y + z) = (x \times y) + (x \times z)$.

De plus, nous supposerons dans toute la suite que V est un corps : ainsi nous pourrons considérer V^n comme un espace vectoriel de dimension n sur le corps V .

¹On a aussi que $S = \text{XOR}_{i=1}^k s_i$.

Nous verrons à la section 3.2 que le fait de considérer V comme un corps impose des restrictions sur son nombre d'éléments, qui doit alors être de la forme $|V| = p^m$ où m est un nombre premier (résultat de base sur les corps finis).

En particulier, on s'intéressera ici surtout aux cas où V est de cardinal 2^d avec d entier.

Pour $|V| \leq 10$, le fait de considérer V comme un corps interdit seulement les cas où le cardinal de V est 6 ou 10. Dans tous les cas, on peut toujours enlever (ou ajouter) à V quelques éléments artificiellement pour que son cardinal soit de la forme p^m avec p nombre premier.

Soit $m \in V^+$ un message à transmettre (on suppose donc le message source préalablement encodé sous forme d'une séquence de chiffres à l'aide d'un code sur V).

Définition 7 Un codage systématique par blocs de k chiffres avec r bits de redondance consiste à :

1. partitionner le message $m = m_0 \dots m_{lk-1} \in V^{lk}$ à transmettre en blocs de k chiffres consécutifs, i.e. :

$$m = M_0 \dots M_{l-1} \text{ avec } M_i = m_{ik} \dots m_{(i+1)k-1}$$

2. Coder chaque bloc M_i en ajoutant r chiffres de redondance $R_i \in V^r$, i.e. :

$$R_i = \phi(M_i)$$

où $\phi : V^k \rightarrow V^r$ est une fonction.

Le message m est alors codé par la séquence : $[M_0 R_0 M_1 R_1 \dots M_{l-1} R_{l-1}]$.

On définit les quantités suivantes :

- $n = k + r$ est appelé *longueur* du code ;
- $R = \frac{k}{n}$ est appelé *rendement* du code ;
- le code est dit *code*(n, k).

Un code (n, k) est dit *t-détecteur* (resp. *t-correcteur*) si il permet de détecter (resp. corriger) toute erreur portant sur t chiffres ou moins.

Exemples :

- L'ajout d'un bit pour contrôler la parité des 7 bits qui le précèdent est un code systématique par bloc. C'est un code (8,7). Il est 1-détecteur et 0-correcteur avec un rendement 87,5%.
- Le contrôle de parité longitudinale et transversale sur 21 bits (avec ajout de 11 bits de contrôle) est un code (32,21). Il est 1-correcteur avec un rendement 65,625%.

3.1.2 Code correcteur et distance de Hamming

Le nombre (minimal) d'erreurs lors de la transmission d'un mot de code est le nombre de chiffres différents entre le mot émis et le mot reçu ; cette quantité définit une distance appelée *distance de Hamming*.

Dans toute la suite on considère un code de longueur n ; les mots de code sont donc des éléments de V^n qui est vu comme un espace vectoriel de dimension n sur V . Soient m_1 et m_2 deux éléments de V^n : $m_1 \oplus m_2$ (resp. $m_1 \otimes m_2$) désigne donc l'élément de V^n obtenu par addition (resp. multiplication) composante par composante de m_1 et m_2 .

Définition 8 Soit $x = (x_1, \dots, x_n) \in V^n$.

On appelle poids de Hamming de x , notée $w(x)$, le nombre de composantes non nulles de x i.e.

$$w(x) = \text{Card}\{i \in \{1, \dots, n\} / x_i \neq 0\}$$

Si $V = \mathbb{Z}/2\mathbb{Z}$, w vérifie $w(x \oplus y) = w(x) + w(y) - 2w(x \otimes y)$ et on a donc l'inégalité triangulaire : $w(x \oplus y) \leq w(x) + w(y)$.

Définition 9 Soient $x = (x_1, \dots, x_n)$ et $y = (y_1, \dots, y_n)$ deux mots de V^n . On appelle distance de Hamming entre x et y , notée $d(x, y)$, le nombre de composantes pour lesquelles x et y diffèrent, i.e.

$$d(x, y) = w(x - y).$$

La quantité d ainsi définie est bien une distance sur V^n ; en effet pour tout x, y et z dans V^n on a :

- $d(x, y) \in \mathbb{R}^+$;
- $d(x, y) = 0 \iff x = y$;
- $d(x, y) = d(y, x)$;
- $d(x, y) \leq d(x, z) + d(y, z)$.

Cette distance de Hamming permet de caractériser le nombre d'erreurs que peut corriger un code C de longueur n . En effet, soit $m \in C$ un mot de n chiffres émis et soit m' le mot reçu, supposé différent de m (i.e. $d(m, m') > 0$) :

- pour que C puisse détecter une erreur, il est nécessaire que $m' \notin C$ (sinon, le mot reçu est un mot de C donc considéré correct).
- pour pouvoir faire une correction (i.e. retrouver m à partir de m' , il faut que m soit l'unique mot de C le plus proche de m' ; i.e.

$$\forall x \in C : x \neq m \implies d(x, m') > d(m, m').$$

Définition 10 Soit C un code de longueur n . C est dit t -correcteur ssi

$$\forall x \in V^n : \text{Card}\{c \in C / d(x, c) \leq t\} \leq 1.$$

La correction d'une erreur peut aussi être décrite en considérant les boules $B_t(c)$ de centre $c \in C$ et de rayon t :

$$B_t(c) = \{x \in V^n / d(c, x) \leq t\}.$$

Lors de la réception de m' , on peut corriger m' en m ssi on a :

$$B_{d(m, m')}(m') \cap C = \{m\}.$$

La capacité de correction d'un code C est donc liée à la distance minimale entre deux éléments de C .

Définition 11 La distance minimale du code C , notée $\delta(C)$, est définie par :

$$\delta(C) = \min_{(c_1, c_2) \in C^2; c_1 \neq c_2} d(c_1, c_2).$$

Ces deux remarques sont à la base du théorème suivant qui caractérise un code t -correcteur.

Théorème 7 Soit C un code de longueur n . Les propriétés suivantes sont équivalentes :

- (i) C est t -correcteur ;
- (ii) $\forall x \in V^n : \text{Card}\{c \in C / d(x, c) \leq t\} \leq 1$;
- (iii) $\forall c_1, c_2 \in C : c_1 \neq c_2 \implies B_t(c_1) \cap B_t(c_2) = \emptyset$;
- (iv) $\forall c_1, c_2 \in C : c_1 \neq c_2 \implies d(c_1, c_2) > 2t$;
- (v) $\delta(C) \geq 2t + 1$.

La preuve est immédiate :

- (i) \iff (ii) est la définition d'un code t -correcteur.
- (ii) \implies (iii) : par la contraposée. Supposons $\exists x \in B_t(c_1) \cap B_t(c_2)$; ainsi c_1 et c_2 sont à une distance $\leq t$ de x . D'après (ii), cela n'est possible que pour au plus un mot de code ; donc $c_1 = c_2$.
- (iii) \implies (iv) : par l'absurde. Si $d(c_1, c_2) \leq 2t$; soient i_1, \dots, i_d avec $d \leq 2t$ les indices où les chiffres de c_1 et c_2 diffèrent. Soit x le mot de V^n dont les chiffres sont les mêmes que ceux de c_1 sauf les chiffres en position $c_1, \dots, c_{d/2}$ qui sont égaux à ceux de c_2 . Alors $d(x, c_1) = d/2 \leq t$ et donc $x \in B_t(c_1)$. De même, $d(x, c_2) = d - (d/2) \leq t$ et donc $x \in B_t(c_2)$. D'où $x \in B_t(c_1) \cap B_t(c_2)$ ce qui contredit (iii).
- (iv) \implies (v) : d'après (iv), on a $d(c_1, c_2) \geq 2t + 1 \forall (c_1, c_2) \in C^2$. D'où $\delta(C) \geq 2t + 1$.
- (v) \implies (ii) : par la contraposée. Supposons $\exists x \in V^n : \text{Card}\{c \in C / d(x, c) \leq t\} \geq 2$. Il existe alors $c_1 \neq c_2$ tels que $d(x, c_1) \leq d(x, c_2) \leq t$. D'où comme d , distance de Hamming, vérifie l'inégalité triangulaire : $d(c_1, c_2) \leq d(c_1, x) + d(x, c_2) \leq 2t$. Donc $\delta(C) \leq 2t$.

cqfd.

Exercice 1. Montrer que si C est t -correcteur alors C est d -détecteur avec $d \geq 2t$. La réciproque est-elle vraie ?

Exercice 2. Soit C un code (n, k) sur un vocabulaire V . Ecrire un programme qui calcule le taux de détection et de correction du code ; donner le coût de ce programme en fonction de k et n .

Application : Soit le code binaire $\{0000000000, 0000011111, 1111100000, 1111111111\}$. Que valent n et k pour ce code ? Calculer son rendement, son taux de détection et de correction.

Exercice 3. Le contrôle de parité transversale et longitudinale sur 4 bits conduit au code $(9,4)$ suivant :

Le mot de code associé à la source $b_0b_1b_2b_3$ est :

$$[b_0, b_1, b_2, b_3, (b_0 + b_1), (b_2 + b_3), (b_0 + b_2), (b_1 + b_3), (b_0 + b_1 + b_2 + b_3)].$$

On obtient ainsi $C = \{000000000, 000101011, 001001101, 001100110, \dots\}$.

1. Montrer que C est exactement 1 correcteur.
2. Donner une configuration comportant 2 erreurs non corrigibles.
3. Montrer que C est 3-détecteur mais pas 4-détecteur.

3.1.3 Code parfait

Un code t -correcteur permet de corriger toute erreur e de poids $w(e) \leq t$; mais il peut subsister des erreurs détectées non corrigées (voir exercice précédent). On dit qu'un code est parfait lorsque toute erreur détectée est corrigée. Ce paragraphe étudie quelques propriétés des codes parfaits.

Définition 12 *Un code t -parfait est un code t -correcteur dans lequel toute erreur détectée est corrigée.*

Si C est t -parfait, lorsqu'on reçoit un mot $m' \notin C$, alors il existe un unique mot m de C tel que $d(m, m') \leq t$. De plus, comme C est t -correcteur, les boules de rayon t et de centre des mots de C sont deux à deux disjointes ; d'où le théorème suivant.

Théorème 8 *Le code $C(n, k)$ sur V est t -parfait ssi les boules de centre les mots de C et de rayon t forment une partition disjointe de V^n , i.e.*

$$V^n = \bigsqcup_{c \in C} B_t(c).$$

Évidemment, une telle partition n'est possible que pour certaines valeurs de n et k , en fonction du cardinal de V . Le théorème suivant donne des conditions nécessaires pour l'existence de codes t -correcteurs et t -parfaits.

Théorème 9 *Soit $C(n, k)$ un code t correcteur sur V . Alors*

$$1 + C_n^1(|V| - 1) + C_n^2(|V| - 1)^2 + \dots + C_n^t(|V| - 1)^t \leq |V|^{n-k}.$$

Si de plus il y a égalité, alors $C(n, k)$ est t -parfait.

La preuve repose sur le calcul du cardinal de $\biguplus_{c \in C} B_t(c)$, les boules étant 2 à 2 disjointes dans le cas d'un code t -correcteur.

En effet, le cardinal d'une boule de rayon t de V^n est :

$$|B_t(x)| = 1 + C_n^1(|V| - 1) + C_n^2(|V| - 1)^2 + \cdots + C_n^t(|V| - 1)^t.$$

Comme un code $C(n, k)$ possède exactement $|V|^k$ éléments de V^n associées à des boules 2 à 2 disjointes dont l'union est incluse dans V^n , on a :

$$|V|^k (1 + C_n^1(|V| - 1) + C_n^2(|V| - 1)^2 + \cdots + C_n^t(|V| - 1)^t) \leq |V|^n,$$

d'où la première inégalité. Si il y a égalité, alors le cardinal de l'union est égal à $|V|^n$; de plus, comme le code est t -correcteur, les boules sont disjointes. Donc leur union est égale à V^n et le code est parfait. cqfd.

Exercice 4. Montrer que tout code 1-correcteur sur des mots de $k = 4$ bits ($V = \{0, 1\}$) requiert au moins 3 bits de redondance. Montrer que si il existe un code 1-correcteur avec 3 bits de redondance, alors il est parfait.

3.1.4 Cas particulier où $V = \{0, 1\}$ - Codes de Hamming

Dans le cas d'un vocabulaire binaire, on peut prendre $V = \mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$.

Soit $x \in F_2^n$; on a $w(x) = \text{Card}\{i \in \{1, \dots, n\} / x_i \neq 0\} = \sum_{i=1}^n x_i$.

Dans F_2 , $a + b = a - b$; on a alors les propriétés suivantes :

Propriété 8 Soit x et y deux éléments quelconques de F_2^n ; on a :

- $w(x) = d(x, 0)$;
- $d(x, y) = w(x + y)$;
- $w(x + y) = w(x) + w(y) - 2w(x.y)$.

Exercice 5. Soit un code binaire $(k + r, k)$ 1-correcteur qui consiste à ajouter r bits de redondance pour k bits de données.

a. Montrer que

$$k \leq 2^r - r - 1.$$

b. En déduire une borne sur le rendement maximal d'un code 1-correcteur dont le nombre de bits de contrôle est 3, puis 4, puis 5, puis 6.

c. Existe-t-il un code 1-parfait de longueur $n = 2^m$?

Codes de Hamming.

Connaissant k , l'inégalité $2^r - r \geq k + 1$ permet de déterminer le nombre minimal de bits de contrôle à ajouter pour obtenir un code 1-correcteur. Les codes de Hamming (1950) permettent alors d'atteindre cette limite théorique : pour n bits, le nombre de bits de contrôle est $\log_2 n + 1$. Ce sont des

codes $(n, n - \lfloor \log_2 n \rfloor - 1)$.

Soit $c = c_1 \dots c_n \in \{0, 1\}^n$ un mot de code. Les bits c_i dont l'indice i est une puissance de 2 sont des bits de contrôle ; les autres sont des bits de données. Le bit de contrôle d'indice $i = 2^l$ est le ou-exclusif (contrôle de parité) de tous les bits de données dont les indices écrits en base 2 ont le l ème bit à 1.

Pour assurer la correction, le contrôle de parité est fait de la façon suivante. Tous les bits de contrôle d'indice $i = 2^l$ sont vérifiés ; une erreur est détectée si l'un de ces bits est erroné (parité erronée). Soit alors e la somme des indices des bits de contrôle i qui ne sont pas vérifiés. Si il y a une seule erreur, elle provient alors du bit e .

Théorème 10 *Le code de Hamming $(n, n - \lfloor \log_2 n \rfloor - 1)$ est un code 1-correcteur qui requiert un nombre de bits de contrôle minimal parmi tous les codes (n, k) qui sont 1-correcteur.*

En particulier, le code de Hamming $(2^m - 1, 2^m - 1 - m)$ est un code 1-parfait.

La preuve découle directement des propriétés ci-dessus. cqfd.

Exemple : Code de Hamming (7,4). Les bits c_1, c_2, c_4 sont des bits de contrôle ; les bits c_3, c_5, c_6, c_7 sont des bits de données. On a : $3 = 1 + 2$; $5 = 1 + 4$; $6 = 2 + 4$; $7 = 1 + 2 + 4$. D'où : $c_1 = c_3 \oplus c_5 \oplus c_7$; $c_2 = c_3 \oplus c_6 \oplus c_7$; enfin $c_4 = c_5 \oplus c_6 \oplus c_7$.

Ainsi 1101 est codé sous la forme : 1010101.

Correction d'erreur dans le code (7,4). En reprenant l'exemple, supposons que l'on reçoit le mot 1111101 : le message source est alors 1101, mais le contrôle de parité indique que les bits 2 et 4 sont erronés ; la correction d'une erreur unique est alors réalisée en modifiant le bit d'indice $4+2=6$. Le mot corrigé est alors 1111 (qui est codé en 1111111).

3.1.5 Codes cycliques : CRC

Pour qu'un code correcteur soit intéressant en pratique, il faut qu'il soit efficace à calculer d'une part (i.e. implémentable sur un circuit simple) et que d'autre part il permette facilement de calculer le mot de code le plus proche du mot reçu lors de la détection d'erreurs.

De plus, il est important de pouvoir construire des codes ayant un rendement maximal : pour un nombre de chiffres de redondance donné, le taux de correction doit être maximal.

En pratique, les codes les plus utilisés qui satisfont à ces critères sont les codes de redondance cyclique, appelés *CRC*. Ces codes sont des codes linéaires (i.e. les bits de contrôle sont des combinaisons linéaires des bits d'information) qui sont de plus stables par décalage des chiffres. Un tel code est caractérisé par un *polynôme générateur* ; on parle parfois de *code polynomial*.

3.2 Bases mathématiques des codes cycliques

Un code correcteur peut être vu comme l'image d'une application ϕ de V^k dans V^n . L'analyse de ϕ dans un contexte quelconque est difficile ; aussi, on se restreint au cas où ϕ est linéaire. Pour pouvoir étudier ce cas, il faut que V^k et V^n soit des espaces vectoriels, donc que V soit un corps.

Tous les résultats d'algèbre linéaire sont alors valides. Par exemple, si ϕ est une application linéaire de V^n , alors : $\dim(\text{Im}(\phi)) + \dim(\text{Ker}(\phi)) = n$.

Dans cette section, nous rappelons les conditions pour que V puisse être muni d'une structure de corps fini. Puis nous rappelons le théorème chinois des restes qui explicite une bijection (plus précisément un isomorphisme d'anneau) entre V^n d'une part et l'ensemble des polynômes à coefficients dans V et de degré strictement inférieur à n d'autre part.

Enfin, nous étudions la factorisation de $X^n - 1$ dans un corps fini qui est à la base de la construction des codes cycliques.

3.2.1 Corps fini

Plus précisément, tout corps fini est de cardinal p^d où p est un nombre premier et d un entier. Il existe alors un unique corps F_q à $q = p^d$ éléments. Ce corps peut être caractérisé algorithmiquement de la façon suivante. p étant un nombre premier, $F_p = Z/pZ$ est un corps. Soit alors $F_p[x]$ l'anneau des polynômes à coefficients dans Z/pZ ; cet anneau est un anneau *euclidien*² et *principal*³ : la notion de pgcd existe et il y a des polynômes *irréductibles*, i.e. premier avec tous les polynômes de degré inférieur.

En particulier, pour tout d , il existe au moins un polynôme $I_d(x) = x^d + \sum_{i=0}^{d-1} \alpha_i x^i$ irréductible de degré d . L'anneau quotient $K = F_p[x]/I_d(x).F_p[x]$ est alors un corps : cette construction est analogue à la construction du corps $Z/q.Z$ avec q nombre premier, mais en remplaçant Z par $F_p[x]$ et l'entier premier q par le polynôme irréductible $I_d(x)$.

Ainsi on a construit un corps dont tous les éléments peuvent être vus comme les restes dans la division euclidienne par le polynôme $I_d(x)$ qui est de degré d . Comme il y a p^d restes possibles, $|K| = p^d$.

Tout corps F_q est alors isomorphe à K ; ceci fournit une caractérisation algorithmique du corps F_q .

3.2.2 Anneau quotient $V[X]/P(X)$

Soit $V[X] = \{P = \sum_{i=0}^d a_i X^i / d \in \mathbb{N}, (a_0, \dots, a_d) \in V^{d+1}\}$ l'anneau des polynômes à coefficients dans V . On note $\deg(P)$ le degré du polynôme P .

²Il existe une division euclidienne, et donc des pgcd et des éléments premiers – ou irréductibles – c'est à dire sans diviseur trivial.

³tout élément admet une décomposition unique en facteurs irréductibles.

Comme V est un corps, $V[X]$ est un anneau euclidien : $\forall A, B \in V[x] : \exists!(Q, R) \in V[x]$ avec $\deg(R) < \deg(B)$ tels que :

$$A = B.Q + R.$$

Q est le polynôme quotient dans la division euclidienne de A par B ; le reste R est aussi noté $A \bmod B$.

Soit alors P un polynôme de degré $d \geq 1$; grâce à la division euclidienne, l'ensemble des polynômes de degré inférieur strictement à d peut être muni d'une structure d'anneau quotient, noté $V[x]/P$ et défini comme suit :

$$V[X]/P = \{A \bmod P : A \in V[X]\}$$

En effet, soit A et B deux polynômes de $V[X]/P$. Les opérations arithmétiques d'addition et de multiplication sont définies comme suit :

$$A +_{V[X]/P} B = (A +_{V[X]} B) \bmod P$$

$$A \times_{V[X]/P} B = (A \times_{V[X]} B) \bmod P$$

Par abus de langage, nous notons dans la suite ces opérations $+$ et \times (ou \cdot) ; elles sont bien sûr toujours réalisées modulo P dans l'anneau $V[X]/P$.

Ainsi, $(V[X]/P, +, \times)$ a bien une structure d'anneau commutatif :

- le polynôme nul ($0.X^0$ noté 0) est élément neutre pour $+$;
- le polynôme unité ($1.X^0$ noté 1) est élément neutre pour \times ;
- $(V[X]/P, +, 0)$ est un groupe commutatif ;
- $(V[X]/P, \times)$ est un monoïde : \times est stable et associative.
- \times est distributive par rapport à $+$;
- \times est commutative ; l'anneau est donc commutatif.

3.2.3 Isomorphisme entre $V[X]/P$ et $V^{\deg(P)}$

Soit P un polynôme de degré n ; l'ensemble $V[X]/P$ est alors celui des polynômes de degré strictement inférieur à n .

A tout vecteur $u = [u_0, \dots, u_{n-1}]$ de V^n , on peut associer de manière bijective le polynôme $\psi(u) = \sum_{i=0}^{n-1} u_i X^i$. De part les propriétés de l'opération modulo, on a trivialement les propriétés suivantes :

- $\forall u, v \in V^n, \forall \lambda \in V : \psi(u +_{V^n} \lambda \cdot_{V^n} v) = \psi(u) +_{V[X]/P} \lambda \times_{V[X]/P} \psi(v)$
- $\psi(0_{V^n}) = 0_{V[X]/P}$

Ainsi, ψ est un isomorphisme entre les deux V -espaces vectoriels V^n et $V[X]/P$.

De plus, nous avons vu que $V[X]/P$ est un anneau ; ψ confère alors à V^n une structure d'anneau. La multiplication \times_{V^n} peut être définie à partir de la multiplication \times dans $V[X]/P$:

$$\forall u, v \in V^n : u \times_{V^n} v = \psi^{-1}(\psi(u) \times \psi(v))$$

Par abus de notation, cette multiplication sera notée \times . Son sens n'est bien sûr définie que par rapport au polynôme P de degré n qui définit l'anneau $V[X]/P$.

3.2.4 PGCD et polynômes irréductibles

De la même manière que l'on peut décomposer un entier $n = p_1^{n_1} \dots p_k^{n_k}$ sous forme de produits et puissances de nombres premiers, on peut décomposer un polynôme sous la forme de produits de polynôme "premiers" (on dit en fait *irréductibles*). Le but de ce paragraphe est d'introduire cette décomposition.

L'anneau $V[X]$ des polynômes à coefficients dans V étant euclidien, l'opération de pgcd est définie ; l'algorithme d'Euclide appliqué à deux polynômes U et V est valide et fournit un polynôme de degré maximal (unique si on le choisit unitaire) qui divise à la fois U et V . Par ailleurs, l'identité de Bezout est valide :

$$\forall u, v \in V[x] \quad \exists a, b \in V[X] \quad / \quad a.u + b.v = \text{pgcd}(u, v)$$

De plus, l'algorithme d'Euclide étendu (ou ses variantes plus efficaces) donne un algorithme qui permet de calculer effectivement deux polynômes a et b dont les degrés respectifs sont strictement inférieurs à ceux de $v/\text{pgcd}(u, v)$ et $u/\text{pgcd}(u, v)$.

Deux polynômes sont dits *premiers entre eux* si leur pgcd est de degré nul, i.e. $\text{pgcd}(U, P) = X^0 = 1$. Autrement dit, U et P n'admettent aucun facteur commun non trivial.

Dans, ce cas, l'identité de Bezout s'écrit :

$$\exists A_U, B \in V[X]/A_U.U + B.P = 1,$$

soit encore $A_U.U \bmod P = 1$: autrement dit U est alors *invertible* dans l'anneau quotient $V[X]/P$ et d'inverse $A_U \bmod P$.

De manière similaire à ce qui se passe pour les entiers avec les nombres *premiers*, il existe des polynômes dits *irréductibles* qui n'ont aucun diviseur non trivial. Autrement dit, un polynôme est irréductible ssi il est premier avec tous les polynômes de degré inférieur à lui-même :

$$P \text{ est irréductible} \iff \forall U \neq 0 \in V[X]/P : \text{pgcd}(U, P) = 1$$

Ainsi, tout élément non nul U de l'anneau quotient $V[X]/P$ admet un inverse A_U dans cet anneau ; autrement dit $V[X]/P$ est un corps.

Par ailleurs, l'anneau des polynômes $V[X]$ à coefficients dans V est *principal* donc *factoriel* : tout élément $P \in V[X]$ admet une décomposition en facteurs irréductibles :

$$P = g_1^{d_1} \dots g_k^{d_k}$$

où les d_i sont des entiers non nuls et les polynômes g_i sont irréductibles. Si P est unitaire⁴, les g_i peuvent être choisis unitaires : la décomposition est alors unique à une permutation d'indice près.

⁴i.e. le coefficient du monôme de plus haut degré de P est égal à 1.

Le théorème fondamental de l'algèbre montre que tout polynôme de degré n a exactement n racines – distinctes ou confondues – dans une extension algébrique suffisante (disons la clôture du corps V , i.e. le plus petit corps qui contient toutes les racines de tous les polynômes irréductibles à coefficient dans V). Soient donc α_i ses n racines ; on a $P = \prod_{i=1}^n (X - \alpha_i)$.

Lorsque une racine apparaît plusieurs fois ($\alpha_i = \alpha_j$), elle est dite *multiple*. Sinon, elle est dite simple.

Lorsque P a toutes ses racines simples⁵, on a $P = g_1 \dots g_n$ car les d_i sont alors tous égaux à 1 ; en effet, $d_i \geq 2$ indique que P a au moins une racine multiple dans la clôture algébrique de V .

En particulier, le polynôme $P = X^n - 1$ a toutes ses racines simples. Il peut donc s'écrire comme un produit de polynômes irréductibles :

$$X^n - 1 = g_1 \dots g_k$$

où les g_i sont irréductibles et tous distincts.

Exemple. Dans $\mathbb{F}_2[X]$:

$$X^3 - 1 = (X - 1).(X^2 + X + 1)$$

On vérifie aisément que $X^2 + X + 1$ est irréductible : il n'est divisible ni par X (0 n'est pas racine) ni par $X + 1$ ($-1=1$ n'est pas racine) qui sont les deux seuls polynômes non triviaux de degré inférieur à lui.

3.2.5 Factorisation de $X^n - 1$: classes cyclotomiques

Les facteurs du polynôme $X^n - 1$ jouent un rôle important pour les codes correcteurs cycliques. Ce paragraphe est consacré à la caractérisation de ces facteurs.

Tout d'abord considérons le cas où n n'est pas premier avec $q = p^m$. On a alors $n = n'.p^\mu$ avec n' premier avec p et donc q . Dans \mathbb{F}_q , on a l'identité remarquable⁶

$$(a + b)^{i.p} = a^{i.p} + b^{i.p}$$

On a alors $X^n - 1 = X^{n'.p^\mu} - 1 = (X^{n'} - 1)^{p^\mu}$. Trouver les facteurs de $X^n - 1$ dans \mathbb{F}_q se ramène donc à trouver ceux de $X^{n'} - 1$ dans le cas où n' est premier avec q .

⁵**Racine simple.** Soit a une racine de P ; on a alors $P(a) = 0$ donc $(X - a)$ divise P . Soit Q le polynôme tel que $P = (X - a).Q$. On dit que a est racine simple de P ssi a n'est pas racine de Q , i.e. $Q(a) \neq 0$.

Sinon, dans le cas où $Q(a) = 0$, on dit que a est racine multiple de P .

⁶Il suffit de développer : $(a + b)^{i.p} = \sum_{j=0}^{i.p} C_{i.p}^j a^j b^{i.p-j}$. Or $C_{i.p}^j$ est multiple de p sauf pour $j = 0$ et $j = i.p$. Comme \mathbb{F}_q est de caractéristique p , $p.x = 0$. Les seuls termes non nuls dans le développement sont donc $a^{i.p}$ et $b^{i.p}$.

Dans ce cas, pour tout corps fini \mathbb{F}_q et pour tout n entier premier avec q , il existe une racine primitive n -ième de l'unité α qui vérifie :

$$X^n - 1 = \prod_{i=0}^{n-1} (X - \alpha^i)$$

En général, α n'appartient pas à \mathbb{F}_q mais à une extension algébrique de ce corps.

Exemple. Dans \mathbb{F}_2 , $X^3 - 1 = (X - 1)(X^2 + X + 1)$.

Comme 0 et 1, seuls éléments de \mathbb{F}_2 ne sont pas racines de $X^2 + X + 1$, ce polynôme admet deux racines distinctes dans une extension de \mathbb{F}_2 ; soit α l'une de ces racines.

On a : $\alpha^3 = 1$, d'où $\alpha^4 + \alpha^2 + 1 = \alpha + \alpha^2 + 1 = 0$. Ainsi, α^2 est l'autre racine de $X^2 + x + 1$. Finalement les 3 racines de $X^3 - 1$ sont : $\alpha, \alpha^2, \alpha^3 = 1$.

Tout diviseur de $X^n - 1$ s'écrit donc sous la forme $\prod_{i \in \Sigma} (X - \alpha^i)$ où Σ est une partie de $\{0, \dots, n-1\}$. Le théorème suivant caractérise les diviseurs de $X^n - 1$ qui sont dans $\mathbb{F}_q[X]$, i.e. dont les coefficients sont dans \mathbb{F}_q .

Théorème 11 Soit $\Sigma \subset \{0, \dots, n-1\}$ une partie de $\mathbb{Z}/n\mathbb{Z}$. Alors $g_\Sigma = \prod_{i \in \Sigma} (X - \alpha^i)$ est un diviseur de $X^n - 1$.

De plus, $g_\Sigma \in \mathbb{F}_q[X]$ ssi Σ est stable par multiplication par q .

Les parties Σ stables par multiplication par q et minimales sont de la forme $\Sigma_i = \{i, i.q, i.q^2, \dots, i.q^{s-1}\}$ où s est le plus petit entier tel que $(q^s.i) \bmod n = i$. Σ_i est appelée *classe cyclotomique* de i relative à q modulo n .

Tout diviseur élémentaire de $X^n - 1$ dans $\mathbb{F}_q[X]$ est associé à une telle classe cyclotomique.

Le calcul des classes cyclotomiques permet alors de décomposer facilement $X^n - 1$ dans $\mathbb{F}_q[X]$ en facteurs irréductibles.

Exemple 1. Reprenons l'exemple de la décomposition de $X^3 - 1$ dans $\mathbb{F}_2[X]$.

Soit α une racine primitive de $X^3 - 1$ (dans une extension algébrique).

Les classes cyclotomiques et les polynômes irréductibles associés sont alors :

- pour $i = 0$: $\Sigma_0 = \{0\}$. D'où $g_{\Sigma_0} = X - \alpha^0 = X - 1$.
- pour $i = 1$: $\Sigma_1 = \{1, 2\}$ (car $4 = 1 \bmod 3$). D'où $g_{\Sigma_1} = (X - \alpha^1)(X - \alpha^2)$.

Exemple 2. Considérons la décomposition de $X^7 - 1$ dans $\mathbb{F}_2[X]$. Les classes cyclotomiques et les polynômes irréductibles associés sont alors :

- pour $i = 0$: $\Sigma_0 = \{0\}$. D'où $g_{\Sigma_0} = X - \alpha^0 = X - 1$.
- pour $i = 1$: $\Sigma_1 = \{1, 2, 4\}$ (car $8 = 1 \bmod 7$). D'où $g_{\Sigma_1} = (X - \alpha^1)(X - \alpha^2)(X - \alpha^4)$.

– pour $i = 3$: $\Sigma_3 = \{3, 6, 5\}$ (car $2.5 = 10 = 3 \pmod{7}$). D'où $g_{\Sigma_3} = (X - \alpha^3)(X - \alpha^5)(X - \alpha^6)$.

En fait, on montre que $\mathbb{F}_2[X]$ n'admet que deux polynômes irréductibles de degré 3 : $(1 + X + X^3)$ et $(1 + X^2 + X^3)$. Ces deux polynômes sont donc g_{Σ_1} et g_{Σ_3} .

3.3 Construction de codes cycliques

3.3.1 Codes linéaires

Nous avons vu qu'un code correcteur (n, k) est caractérisé par une application $\phi : V^k \rightarrow V^n$ qui donne le mot de code $c_i \in V^n$ associé à un mot $x \in V^k$. Le code est dit linéaire ssi ϕ est linéaire.

Définition 13 Soit $C = \{\phi(x)/x \in V^k\} \subset V^n$ un code (n, k) sur V . Le code C est dit linéaire ssi ϕ est une application linéaire de $V^k \rightarrow V^n$.

ϕ est alors caractérisé par une matrice G_C comportant k lignes et n colonnes et à coefficients dans V . La matrice G_C est appelée matrice génératrice du code C .

Les lignes de G_C forment un système générateur de C . De plus, comme le codage est injectif, $\text{rang}(G_C) = \dim(C) = k$. Ainsi les k lignes de G_C forment une base du sous-espace vectoriel C , i.e. un système libre et générateur.

Dans la littérature, les mots de code sont représentés sous forme de vecteurs lignes : d'où le choix de représenter ϕ par une matrice rectangulaire avec plus de colonnes que de lignes.

Exemple. Considérons le code $(4, 3)$ de parité sur $V = \{0, 1\}$. Le mot de code associé au mot d'information $X = [x_0, x_1, x_2]$ est alors $B = [b_0, b_1, b_2, b_3, b_4]$ défini par :

$$\begin{cases} b_0 = x_0 \\ b_1 = x_1 \\ b_2 = x_2 \\ b_3 = x_0 + x_1 + x_2 \pmod{2} \end{cases}$$

Ce code est donc un code linéaire sur $V = \mathbb{Z}/2\mathbb{Z}$ de matrice génératrice :

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

On a alors : $B = X.G$

Théorème 12 Soit C un code (n, k) linéaire et soit $d = \min_{x \in C; x \neq 0} w(x)$. Alors

$$\delta(C) = d,$$

i.e. C est $(d-1)$ -détecteur et $\lfloor \frac{d-1}{2} \rfloor$ -correcteur. Le code C est alors dit code (n, k, d) .

Preuve. Comme ϕ est linéaire, $C = \text{Im}(\phi)$ est un sous-espace vectoriel de V^n .

Donc $0 \in C$ (0 est le vecteur dont les n composantes sont nulles). D'où : $\forall c \in C, \delta(C) \leq d(c, 0) = w(c)$, i.e. $\delta(C) \leq \min_{c \in C} w(c)$.

Réciproquement, soient c_1 et c_2 deux éléments de C tels que $\delta(C) = d(c_1, c_2) = w(c_1 - c_2)$. Comme C est un sous-espace vectoriel, $c = c_1 - c_2$ appartient à C ; $\delta(C) \geq \min_{c \in C} w(c)$.

Finalement $\delta(C) = \min_{c \in C} w(c)$. cqfd

Théorème 13 Borne de Singleton. *La distance minimale d d'un code (n, k) linéaire sur V est majorée par :*

$$d \leq n - k + 1$$

Preuve. Soit C un code linéaire défini par l'application linéaire ϕ . Comme ϕ est injective, $\text{rang}(\phi) = k$. D'où $\dim(C) = \dim(\text{Im}(\phi)) = k$.

Considérons le sous-espace E de V^n dont les $k - 1$ dernières composantes sont nulles : $\dim(E) = n - k + 1$.

Ainsi, E et C sont deux sev de V^n et $\dim(C) + \dim(E) = n + 1 > n$. Il existe donc un élément non nul a dans $C \cap E$. Comme $a \in E$, $w(a) \leq n - k + 1$; comme $a \in C$, $\delta(C) \leq w(a) \leq n - k + 1$. cqfd.

Exemple. Le code de Hamming (7, 4) est un code linéaire de matrice génératrice :

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

Sa distance est 3 ; c'est donc un code linéaire (7, 4, 3).

Plus généralement, le code de Hamming pour n bits est un code⁷ $(n, n - 1 - \lfloor \log_2 n \rfloor, 3)$.

3.3.2 Codes cycliques

Un code cyclique est un code linéaire qui est stable pour l'opération de décalage de chiffre.

Définition 14 *On appelle opération de décalage l'application linéaire σ de V^n définie par*

$$\sigma([u_0, \dots, u_{n-1}]) = [u_{n-1}, u_0, \dots, u_{n-2}]$$

⁷nous avons vu qu'un tel code est 1-correcteur

On vérifie trivialement que l'opération σ est linéaire en exhibant sa matrice Σ :

$$\Sigma = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 1 \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

Définition 15 Soit C un code linéaire de V^n . C est cyclique ssi

$$\sigma(C) = C$$

Exemple. Le code de parité $(n, n-1)$ est un code cyclique. En effet, si $c = (c_0, \dots, c_{n-1})$ est un mot de code, alors $c_{n-1} = \sum_{i=0}^{n-2} c_i \pmod 2$. Mais on a alors aussi $c_{n-2} = c_{n-1} + \sum_{i=0}^{n-3} c_i \pmod 2$; ainsi, $\sigma(c) = (c_{n-1}, c_0, \dots, c_{n-2})$ est aussi un mot de code. Le code de parité est donc cyclique.

3.3.3 Construction d'un code cyclique : polynôme générateur

Nous avons vu (§3.2.3) que tout élément $U = [u_0, \dots, u_{n-1}]$ de V^n pouvait être représenté par le polynôme de degré n de $V[X]$:

$$P_U = \sum_{i=0}^{n-1} u_i X^i.$$

Comme $X^n - 1$ est un polynôme de degré n , $V^n \cong V[X]/(X^n - 1)$. Dans $V[X]/(X^n - 1)$, on a alors

$$P_{\sigma(U)} = [X.P_U(X) - u_{n-1} \cdot (X^n - 1)] \pmod{(X^n - 1)} = X.P_U$$

Autrement dit, l'opération de décalage σ d'un vecteur revient à multiplier son polynôme associé par X dans $V[X]/(X^n - 1)$.

Cette propriété est à la base du théorème suivant qui donne une caractérisation algébrique d'un code cyclique.

Théorème 14 Tout code cyclique $C = (n, k)$ admet une matrice génératrice G_C de la forme :

$$G_C = \begin{bmatrix} m \\ \sigma(m) \\ \vdots \\ \sigma^{k-1}(m) \end{bmatrix}$$

avec $m = [a_0, a_1, \dots, a_{n-k} = 1, 0, \dots, 0]$ tel que

$$g(X) = \sum_{i=0}^{n-k} a_i X^i$$

est un diviseur unitaire de $(X^n - 1)$ de degré $r = n - k$.

Le polynôme g est appelé polynôme générateur du code cyclique.

Réciproquement, tout diviseur g de $(X^n - 1)$ est polynôme générateur d'un code cyclique.

Cette propriété permet la construction directe de code correcteur par la donnée d'un polynôme diviseur de $X^n - 1$. Un tel polynôme peut être calculé à partir des classes cyclotomiques relatives à q modulo n .

Exemple : construction d'un code cyclique (7,4) . Nous avons vu que $(X^7 - 1) = (X - 1)(1 + X^2 + X^3)(1 + X + X^3)$ dans $\mathbb{F}_2[X]$. $g = (1 + X^2 + X^3)$ est donc le polynôme générateur d'un code cyclique. Comme g est de degré 3 et que $n = 7$, ce code est un code (7,4). Sa matrice est :

$$G_C = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Plus généralement, la factorisation de $X^7 - 1$ permet d'exhiber 6 diviseurs non triviaux de $X^7 - 1$ qui définissent chacun un code cyclique :

- $g_1 = X - 1$: code cyclique (7,6)
- $g_2 = X^3 + X + 1$: code cyclique (7,4)
- $g_3 = X^3 + X^2 + 1$: code cyclique (7,4)
- $g_4 = (X - 1)(X^3 + X + 1)$: code cyclique (7,3)
- $g_5 = (X - 1)(X^3 + X^2 + 1)$: code cyclique (7,3)
- $g_6 = (X^3 + X + 1)(X^3 + X^2 + 1)$: code cyclique (7,1)

3.3.4 Codage et décodage d'un code cyclique

Le lien entre code cyclique et polynôme générateur est important ; il est à la base des algorithmes efficaces de codage et de décodage qui évitent la multiplication par la matrice génératrice G_C (de coût $O(nk)$) en la remplaçant par une multiplication par le polynôme g , de coût $O(n \log n)$. L'intérêt pratique est que les circuits associés au codage et décodage sont relativement simples à réaliser.

Soit C un code cyclique de polynôme générateur g ; soit G une matrice génératrice de C construite à partir de g comme dans le théorème 14.

Opération de codage.

Soit $a = [a_0, \dots, a_{k-1}] \in V^k$ un mot source et $P_a = \sum_{i=0}^{k-1} a_i X^i$ son polynôme associé.

Le mot de code associé à a est $\phi(a) = aG$ de polynôme associé P_{aG} . De part

l'écriture de G à partir des coefficients de $g(X)$ (théorème 14), on a :

$$\begin{aligned} P_{aG} &= \sum_{i=0}^{k-1} a_i(X^i g(X) \bmod X^n - 1) \\ &= [g(X) \left(\sum_{i=0}^{k-1} a_i X^i \right)] \bmod X^n - 1 \\ &= [g(X) \cdot P_a(X)] \bmod X^n - 1 \end{aligned}$$

Le codage correspond donc à un produit de polynômes, les degrés des monômes étant pris modulo n : en effet, calculer $X^n - 1$ revient à considérer que $X^n = 1 = X^0$

Exemple. Soit le code cyclique $(7,4)$ précédent associé au polynôme $g = (1 + X^2 + X^3)$. Soit $a = [a_0, a_1, a_2, a_3]$ un mot source ; le code de ce mot est alors égal à $s.G$; il est obtenu par multiplication à droite par la matrice G . Le même mot est obtenu en calculant les coefficients du polynôme :

$$(a_0 + a_1X + a_2X^2 + a_3X^3)(1 + X^2 + X^3) \bmod X^7 - 1$$

Ainsi, pour $a = [1001]$: $P_a.g \bmod X^7 - 1 = (1 + X^3)(1 + X^2 + X^3) = 1 + X^2 + 2.X^3 + X^5 + X^6 = 1 + X^2 + X^5 + X^6$. Donc $\phi(a) = [1010011]$.

Détection d'erreur et opération de décodage.

Le codage précédent montre que tout mot de code reçu est un multiple de $g(X)$. Soit $m \in V^n$ un message reçu et soit P_m son polynôme associé. Lors du décodage, on calcule d'abord $P_e = P_m \bmod g$;

- si $P_e = 0$: le polynôme reçu est bien un multiple de g : il correspond donc à un mot de code. On retrouve le message émis en calculant la division de polynôme P_m/g .
- sinon, $P_e \neq 0$: le mot reçu n'est pas un mot de code et il y a donc eu des erreurs lors de la transmission.

La détection d'erreur est donc réalisée via le calcul de $P_e = P_m$ modulo g ; P_e est appelé *syndrome d'erreur*.

Dans le cas où P_e est non nul, on peut procéder à une correction. Une manière brutale est de tabuler tous les mots de V^n en associant à chacun le mot de code le plus proche, qui correspond au message corrigé. La correction se fait alors par lecture de la table.

Le lien entre code cyclique et polynômes fait qu'il existe des algorithmes moins coûteux en place mémoire pour corriger un message erroné. La méthode de Meggitt et ses variantes en sont un exemple.

3.3.5 Classes cyclotomiques et distance minimale

Le taux de correction d'un code cyclique est difficile à calculer. Cependant, ce paragraphe présente un théorème qui permet de garantir une minoration de la distance minimale d'un code, et par suite (cf théorème 12) une minoration du taux de détection. Ce théorème est de plus constructif : il permet la construction de codes cycliques ayant un taux de correction garanti.

On suppose que n est premier avec q . Soit alors α une racine primitive de $X^n - 1$ dans $\mathbb{F}_q[X]$; soit C un code cyclique de polynôme générateur g . Comme g est un diviseur de $X^n - 1$, on a $g = \prod_{i \in \Sigma} (X - \alpha^i)$ où $\Sigma \subset \{0, 1, \dots, n-1\}$ est la réunion de classes cyclotomiques relatives à q modulo n . Le théorème suivant montre que l'analyse de Σ permet d'avoir une minoration de la distance minimale $\delta(C)$ du code C .

Théorème 15 *On suppose que n est premier avec q .*

Si il existe un entier a tel que $\{a+1, a+2, \dots, a+s\} \subset \Sigma$ alors

$$\delta(C) \geq s + 1.$$

Le code C est donc au moins s -détecteur et $\lfloor s/2 \rfloor$ -correcteur.

Exemple. Considérons $n = 7$ et $V = \mathbb{F}_2$. Un code cyclique est alors associé à un diviseur de $X^7 - 1$. Nous avons vu que les classes cyclotomiques relatives à 2 modulo 7 sont :

- $\Sigma_0 = \{0\}$; $g_0 = X - 1$.
- $\Sigma_1 = \{1, 2, 4\}$; $g_1 = (1 + X + X^3)$.
- $\Sigma_3 = \{3, 6, 5\}$; $g_2 = (1 + X^2 + X^3)$.

On a : $\{1, 2\} \subset \Sigma_1$. Ici, $s = 2$: le code (7,4) associé à g_1 est donc au moins 1-correcteur.

De la même façon, on a $\{5, 6\} \subset \Sigma_3$; le code (7,4) associé à g_2 est donc aussi au moins 1-correcteur (c'est un code de Hamming).

Considérons maintenant le code C associé au polynôme $g = g_1.g_2$; comme g est de degré 6, C est un code (7,1) qui comporte 6 bits de redondance. Il est caractérisé par la classe $\Sigma = \Sigma_1 \cap \Sigma_3 = \{1, 2, 3, 4, 5, 6\}$; ici, $s = 6$. Ce code est donc au moins 3-correcteur.

3.3.6 Codes BCH

En application du théorème 15, Bose, Chaudhuri et Hocquenghem ont proposé une méthode de construction de codes cycliques ayant un taux de correction arbitraire. Ces codes sont appelés codes BCH.

Soit a un entier arbitraire. Pour avoir un taux de correction de t au moins, il suffit en effet d'après la proposition précédente que le polynôme g ait parmi ses racines $\{\alpha^{a+1}, \alpha^{a+2}, \dots, \alpha^{a+2t}\}$. Autrement dit, il suffit que g soit associé à la partie $\Sigma \subset \{1, 2, \dots, n-1\}$ plus petite union possible de classes cyclotomiques contenant $\{a+1, a+2, \dots, a+2t\}$.

Un tel code est appelé code BCH. Sa distance minimale est supérieure à $2t$ et donc son taux de correction est garanti supérieur à t . De part leurs propriétés algébriques, il existe des algorithmes spécifiques de codage et décodage des code BCH, plus performants que ceux pour les codes cycliques quelconques.

Dans la plupart des cas pratiques, on choisit n premier avec q ; $X^n - 1$ se factorise alors à partir des classes cyclotomiques, toutes les racines étant de multiplicité 1 (cf §3.2.5). En particulier, les codes BCH les plus utilisés correspondent à $n = q^t - 1$; un tel code est dit *primitif*.

Les paragraphes suivants présentent différents codes de type BCH qui sont utilisés dans l'industrie.

3.4 Codes cycliques usuels

Nous présentons dans cette section rapidement quelques codes utilisés en pratique.

Pour les applications où le taux d'erreurs est faible, un codage de Hamming peut être utilisé. Le Minitel par exemple utilise un code de Hamming (128,120) qui est 1-correcteur : le message est tronçonné en blocs de 15 octets, i.e. 120 bits; Un 16ème octet contient 8 bits de contrôle qui permet de localiser un bit d'erreur parmi les 128. En outre un 17ème octet, dit de validation et ne contenant que des 0, est utilisé pour détecter des perturbations importantes. Le code final est donc un code binaire 1-correcteur de paramètres (136,120).

Cependant, les développements des codes cycliques et notamment la mise au point de procédure de codage et de décodage particulièrement efficaces ont motivé l'intégration de ces codes correcteurs dans de nombreuses applications où le taux d'erreurs est important et donc la capacité de correction critique. Les paragraphes suivants montrent quelques uns de ces codes utilisés pour la lecture de disques compacts et pour la transmission d'images par satellites.

3.4.1 Codes de Reed-Solomon

Les codes de Reed-Solomon sont des codes BCH primitifs sur $V = \mathbb{F}_{2^m}$ (i.e. $q = 2^m$) et de longueur $n = 2^m - 1$ (i.e. $n = q - 1$; n est donc premier avec q).

L'intérêt de ces codes est double. D'une part, ils sont optimaux dans le sens où ils requièrent un nombre de chiffres de redondance minimal pour une capacité de correction fixée. D'autre part, ils sont particulièrement faciles à coder et décoder.

En effet, le polynôme $X^{2^m-1} - 1$ se factorise très simplement sur \mathbb{F}_{2^m} :

ses racines sont tous les éléments non nuls de \mathbb{F}_{2^m} :

$$X^{2^m-1} - 1 = \prod_{\lambda \in \mathbb{F}_{2^m} - \{0\}} (X - \lambda).$$

Le groupe multiplicatif $\mathbb{F}_{2^m} - \{0\}$ étant cyclique, soit α un élément primitif.

On a alors :

$$X^{2^m-1} - 1 = \prod_{i=1}^{2^m-1} (X - \alpha^i).$$

La construction d'un code de Reed-Solomon avec r chiffres de redondance repose sur le choix d'un polynôme générateur de degré r , dont les racines sont consécutives :

$$g(X) = \prod_{i=s}^{s+r-1} (X - \alpha^i).$$

Le code de Reed-Solomon ainsi obtenu est donc un code ($n = 2^m - 1, k = n - r$) avec r arbitraire.

Le taux de correction de ce code est optimal. En effet, en tant que code BCH, la distance minimale δ du code de Reed-Solomon est au moins $\delta \geq r+1$. Or, la borne de Singleton (théorème 13) montre que $\delta \leq n - k + 1 = r + 1$; ainsi, la distance est $\delta = r + 1$ et atteint la borne de Singleton.

Les codes de Reed-Solomon sont très utilisés en pratique. Ainsi, le satellite d'exploration de Jupiter *Galileo* utilise le code de Reed-Solomon (255,223) de distance 33. Ce code est 16-correcteur sur le corps \mathbb{F}_8 . Le corps \mathbb{F}_8 est engendré par le polynôme irréductible de $\mathbb{F}_2[X] : x^8 + x^7 + x^2 + x + 1$. Soit α une racine de ce polynôme. Le polynôme générateur du code de Reed-Solomon (255,223,33) est alors :

$$g = \prod_{j=12}^{43} (X - \alpha^{11j})$$

qui est de degré $r = 32$.

3.4.2 Codes C.I.R.C.

Les codes de Reed-Solomon sont à la base des codes C.I.R.C. (*Cross Interleaved Reed-Solomon Code*, i.e. code de Reed-Solomon à entrelacement croisé). Les codes C.I.R.C. sont des codes "raccourcis" des codes de Reed-Solomon : ils ne contiennent que les mots du code de Reed-Solomon commençant par un nombre fixé de 0, en supprimant ces 0 de tête. Ils ont donc la même distance que le code de Reed-Solomon dont ils sont issus.

Par exemple, le code C.I.R.C (32,28,5) est un code raccourci du code de Reed-Solomon(255,251,5) est utilisé pour les disques compacts. Ce code permet de détecter 4 erreurs et d'en corriger 2, ce qui s'avère assez efficace pour les disques compacts qui sont soumis à de multiples perturbations (rayures, poussières, défauts de surface etc).

3.4.3 Quelques autres codes cycliques

De nombreux autres codes cycliques ou leurs variantes sont utilisés en pratique.

Les codes de Golay sont des codes cycliques dont la longueur n est un nombre premier, de plus premier avec $q = \text{card}(V)$. Dans, ce cas $(X^n - 1)$ possède une factorisation simple en deux facteurs non triviaux de degré chacun $(n - 1)/2$. Ainsi, le code de Golay avec $n = 11$ et $q = 3$ est un code ternaire associé à un polynôme générateur de degré $r = 5$. Ce code a donc pour paramètre $(11, 6)$; il est 2-correcteur et parfait.

Les codes de Reed-Muller sont des codes (n, k) cycliques *étendus* : les mots de code sont ceux d'un code cyclique $(n - 1, k)$ avec un chiffre supplémentaire qui est la somme des $n - 1$ chiffres du mot du code cyclique. Ces codes de Reed-Muller sont notamment utilisés pour la transmission d'images par satellite.

Chapitre 4

Méthodes de chiffrement

4.1 Introduction

Chiffrer un texte consiste à le transformer en une forme codée par chiffrement et réciproquement par déchiffrement. Ces transformations sont généralement paramétrées par une ou plusieurs clés. La motivation principale du chiffrement réside en la sécurisation d'un canal de communication qui peut être espionné.

Les trois services principaux fournis par les systèmes de chiffrement sont :

Secret Le secret consiste à *empêcher l'accès* aux informations qui transitent pour ceux qui ne sont pas autorisés. Ils peuvent lire ce qui passe sur le canal mais ne peuvent pas le déchiffrer.

Authentification consiste à *signer* électroniquement un document afin de prouver en être l'auteur et qu'il ne s'agit donc pas un faux.

Intégrité permet de vérifier que le message n'a pas subi d'*altérations* lors de son parcours. Cette vérification ne se place pas au même niveau que celles vues au chapitre précédent, elle concerne plutôt une modification volontaire par un tiers lors de son transfert sur le canal, qui aurait donc dans ce cas modifié les *checksums* pour masquer ses modifications.

Le chiffrement classique ne s'intéresse généralement qu'au premier aspect et, jusqu'à ces dernières années, ne s'intéressait qu'aux clés secrètes. Les 20 dernières années ont vu émerger l'étude de nouvelles tendances :

- l'authentification devient aussi, voire même plus, importante que le secret¹,
- une partie de la clé ne doit pas être privée, afin de ne pas provoquer une explosion du nombre de clés nécessaires pour communiquer avec un grand nombre de personnes.

¹Cela est particulièrement vraie dans le cas du commerce électronique : il faut pouvoir prouver que la commande vient bien de la personne à qui la livraison est destinée pour éviter les contestations.

De plus, les opérations de chiffrement et déchiffrement portant sur de grandes quantités de données, le critère d'efficacité est très important afin de pouvoir chiffrer "à la volée" des flux audio ou vidéo par exemple.

Un système de chiffrement idéal devrait donc résoudre tous ces problèmes en même temps : utiliser des clés publiques, fournir du secret, de l'authentification et de l'intégrité. Malheureusement, il n'existe pas de technique unique qui satisfasse ces trois critères. Les systèmes conventionnels comme le DES utilisent des clés privées et les systèmes à clé publique fournissent de l'authentification mais sont inefficaces pour le chiffrement de grandes quantités de données car trop coûteux. Cependant les systèmes conventionnels et les systèmes à clé publique ne sont pas mutuellement exclusifs ; ils se complètent au contraire très bien.

Nous allons dans la suite présenter différents algorithmes de chiffrement. La notation utilisée est la suivante : E et D représentent respectivement les opérations de chiffrement et de déchiffrement. Il est requis que $D(E(M))=M$. On suppose toujours que D est secret mais E peut être public.

4.1.1 Méthode d'attaque

Satisfaire le secret signifie qu'un attaquant ne peut pas :

- trouver M à partir de E(M) ; le système de chiffrement doit être résistant aux attaques sur le message codé,
- trouver la méthode de déchiffrement D à partir d'une séquence $\{E(M_i)\}$ pour une séquence quelconque de messages en clairs $\{M_1, M_2, M_3, \dots\}$, le système doit être sûr vis-à-vis des attaques avec du texte en clair.

En liaison avec la première partie du cours on peut dire qu'une clé est bonne si son entropie est élevée car cela signifie qu'elle ne contient pas de patterns répétés plusieurs fois.

Attaque brutale

Elle consiste en l'énumération de toutes les valeurs possibles de la clé, c'est-à-dire à une exploration complète de l'espace des clés. La complexité de cette méthode apparaît immédiatement : une clé de 128 bits oblige à essayer 2^{128} combinaisons différentes.

Pour une clé de 64 bits, il existe $1.844 * 10^{19}$ combinaisons différentes, sur un ordinateur calculant un milliard de clés par seconde il faudra 584 ans pour être sûr de trouver la clé².

Attaque par séquences connues

Ce type d'attaque consiste à supposer une certaine partie du message (par exemple les entêtes standards dans le cas d'un message transmis par courrier

²ou un an avec 584 ordinateurs tournant en parallèle

électronique) et de partir de cette connaissance pour essayer de deviner la clé.

Elle peut marcher si l'algorithme de chiffrement laisse apparaître des patterns du message original.

Attaque par séquences forcées

Cette méthode d'attaque est basée sur la précédente, elle consiste à faire chiffrer par la victime un bloc dont l'attaquant connaît le contenu.

Attaque par analyse différentielle

Cette attaque se base utilise les faibles différences existant entre des messages successifs (par exemple des logs de serveur) pour essayer de deviner la clé.

4.2 Systèmes conventionnels à clé privée

Dans un système cryptographique conventionnel, E et D sont paramétrés par une simple clé K, on a donc $D_K(E_K(M)) = M$. Typiquement la méthode pour passer à D_K et E_K à partir de K sont publiques, seule K est donc secret. Avec une telle méthode les propriétés de secret et d'authentification sont assurées : si les deux parties partagent le secret K, la clé, ils peuvent s'envoyer des messages qui sont privés (car un espion ne peut pas calculer ou deviner $D_K(C)$) et authentifiés, car un imposteur ne peut pas calculer $E_K(C)$. Cette approche n'assure cependant pas la propriété d'intégrité, il est généralement adjoint au message chiffré une forme compressée de ce dernier pour assurer cette dernière propriété.

Nous allons voir dans la suite deux systèmes classiques de chiffrement à clé publique : le DES et l'exponentiation.

4.2.1 Le DES

Le système de chiffrement traditionnel à clé privée le plus connu est le DES (Data Encryption Standard). Il est très bien documenté et ne sera pas présenté en détails ici, il n'est donné qu'à titre de comparaison de son fonctionnement avec celui des autres systèmes présentés.

Ce système fonctionne par blocs, il travaille sur des blocs de 64 bits en utilisant une clé de 56 bits. Le même algorithme est utilisé pour chiffrer et déchiffrer. La transformation employée peut s'écrire sous la forme $P^{-1}(F(P(M)))$, où P est une permutation et F une fonction combinant permutation et substitution.

Le gros avantage du DES est qu'il repose, tant pour le codage que pour le décodage, sur des opérations facilement implantables au niveau matériel,

il est donc possible d'obtenir des taux de chiffrement très élevés, de l'ordre de 40Mo/s il y a une dizaine d'années, avec du matériel spécifique.

La sûreté du DES est produite de manière classique en alternant substitutions et permutations. La fonction F est obtenue en cascasant une fonction $f(x, y)$. où x fait 32 bits et y 48. Chaque étape de la cascade est appelée un tour. Une séquence de chaînes de 48 bits $\{K_i\}$ est générée à partir de la clé K . Notons $L(x)$ et $R(x)$ les parties gauche et droite de x , alors si M_i est le résultat du tour i , on a :

$$L(M_i) = R(M_{i-1}) \quad (4.1)$$

et

$$R(M_i) = L(M_{i-1}) \text{ XOR } f(L(M_i), K_i) \quad (4.2)$$

Le secret repose sur le fait qu'après 16 tours le résultat sera statistiquement "plat", c'est-à-dire que les caractéristiques générales du message source (la fréquence des caractères, le nombre d'espaces, ...) seront indétectables. De plus il dispose d'une caractéristique très importante pour éviter les attaques par analyse différentielle : une légère modification de la clé ou du texte à chiffrer provoque des changements importants dans le texte chiffré.

La plupart des systèmes de chiffrement "classiques" c'est-à-dire vieux de plus de 30 ans, sont basés sur l'utilisation de permutations, substitutions et d'opérations logiques ou numériques entre la clé et le message ainsi obtenu. Plus récemment sont apparues des méthodes mettant en jeu des propriétés sur les nombres par opposition à la séquence de bits. Nous allons présenter la plus connue : l'exponentiation. Elle nous permettra d'introduire certaines des propriétés utilisées également pour la cryptographie à clé publique.

4.2.2 L'exponentiation

Pohlig et Hellman en 1978 ont construit un autre type de chiffreur qui n'est pas basé sur les méthodes classiques de substitution et transposition. Leur technique est simple : on pose $p > 2$ premier et K une clé dans $\{1, p-1\}$ avec $PGCD(K, p-1) = 1$ c'est-à-dire K et $p-1$ premiers entre eux. Si M est le message à chiffrer dans $\{1, p-1\}$ une fonction de chiffrement E peut être :

$$E(M) = M^K \text{ mod } p \quad (4.3)$$

Il reste à trouver la fonction de décodage. Pour cela, nous allons utiliser le théorème suivant :

Théorème 16 Notons Z_n^* l'ensemble des nombres premiers avec n . On a :
 $Z_n^* = \{x \in Z_n : x > 0 \text{ et } PGCD(x, n) = 1\}$

et

$$a * Z_n^* = \{a * x \pmod n \mid x \in Z_n^*\}$$

Exemple

$$Z_{10}^* = \{1, 3, 7, 9\}$$

$$2 * Z_{10}^* = \{2, 6, 14, 18\} \pmod{10} = \{2, 6, 4, 8\}$$

$$3 * Z_{10}^* = \{3, 9, 21, 27\} \pmod{10} = Z_{10}^*$$

On pose $n > 1$ et $a \in Z_n^*$. On a alors :

i Pour tout x et y : $a * x \equiv a * y \pmod n$ ssi $x \equiv y \pmod n$

ii $a * Z_n^* = Z_n^*$

iii a a un inverse pour la multiplication modulo n ; i.e. il existe $b \in Z_n^*$ tel que $b * a \equiv 1 \pmod n$

Preuve

Si $x \equiv y \pmod n$ alors on a trivialement $a * x \equiv a * y \pmod n$. Réciproquement, on a $a * x \equiv a * y \pmod n$ donc $a * (x - y) \equiv 0 \pmod n$ et donc n est un diviseur de $a * (x - y)$. Comme $\text{PGCD}(a, n) = 1$ n divise $(x - y)$ et donc $x \equiv y \pmod n$ d'où la proposition i.

Pour ii. Si $x \in Z_n^*$ on a $\text{PGCD}(x, n) = 1$ et donc $\text{PGCD}(a * x, n) = 1$. On a donc $a * x \pmod n \in Z_n^*$. $a * Z_n^*$ est inclus dans Z_n^* . De i. on sait que si $a * x \equiv a * y \pmod n$ alors $x \equiv y \pmod n$. Donc $a * Z_n^*$ contient $n - 1$ éléments distincts et n'est donc pas un sous-ensemble de Z_n^* , d'où la proposition ii. En particulier, comme $1 \in Z_n^*$, il existe un $b \in Z_n^*$ tel que $a * b \pmod n = 1$ d'où la proposition iii.

La condition $\text{PGCD}(K, p - 1) = 1$ implique, d'après le théorème 16, qu'il existe I tel que $I * K \equiv 1 \pmod{p - 1}$ et que I peut être calculé. Il est important de noter que I n'est pas une autre clé : I peut être dérivé de K et vice-versa de manière directe. On peut alors construire la fonction de déchiffrement :

$$D(C) = C^I \pmod p \tag{4.4}$$

On peut alors montrer que la condition $D(E(M)) = M$ est satisfaite, pour cela on doit utiliser le théorème de Fermat présenté un peu plus loin en section 4.3.2. De plus $E(D(M)) = M$ également, ce qui signifie que les fonctions D et E sont inverses, il est possible de chiffrer indifféremment avec D ou E , ce qui est aussi utile pour l'authentification, comme nous le verrons dans la suite. Il s'agit cependant d'un système traditionnel à clé secrète car, comme I peut être facilement calculé à partir de K (en utilisant l'algorithme d'Euclide étendu voir plus bas), le secret réside dans K uniquement. Nous verrons dans la suite que si p est le produit de deux grands nombres premiers, trouver I à partir de K est non trivial, ce qui permet de diviser la clé en deux parties.

Algorithme d'Euclide étendu

L'algorithme d'Euclide permet de calculer le PGCD de deux nombres entiers. Le principe est simple : si a et b sont deux entiers avec $b < a$ et ont un diviseur commun d alors $a - b, a - 2b, \dots$ sont aussi divisibles par d .

Si $a - nb = 0$ cela signifie que a est divisible par b et le problème est donc résolu. Pour itérer l'algorithme on prend n comme quotient entier de la division de a par b , $a - nb$ en est alors le reste et on itère le procédé.

Exemple : Déterminons le PGCD de 522 et 453. On calcule successivement :

$$\begin{array}{ll}
 \text{(a)} & 522 \\
 \text{(b)} & 453 \\
 \text{(c)} & 522-453=69 \quad (\text{a-b}) \\
 \text{(d)} & 453-6*69=39 \quad (\text{b-6c}) \\
 \text{(e)} & 69-39=30 \quad (\text{c-d}) \\
 \text{(f)} & 39-30=9 \quad (\text{d-e}) \\
 \text{(g)} & 30-27=3 \quad (\text{e-3f}) \\
 \text{(h)} & 9-9=0 \quad (\text{f-3g})
 \end{array}$$

L'obtention de la valeur 0 arrête l'algorithme et 3 est donc le PGCD de 522 et 453.

L'algorithme d'Euclide étendu consiste à utiliser l'algorithme d'Euclide pour trouver, par divisions successives, d tel que $a * d = 1 \pmod{n}$. Il fonctionne exactement sur le même principe, mais s'arrête lorsque le résultat obtenu est 1 et non 0. Les deux valeurs initiales sont n et a . On conserve en plus la valeur du coefficient obtenu après chaque opération, cette valeur est le d cherché lors de l'arrêt de l'algorithme. Au début, il est égal à 0 et 1 pour les deux premières lignes correspondant à n et a . Par exemple, si on prend $n = 2668$, $a = 17$ et que l'on écrit la valeur du coefficient dans la colonne de droite, on obtient :

$$\begin{array}{llll}
 \text{(a)} & 2668 & & 0 \\
 \text{(b)} & 17 & & 1 \\
 \text{(c)} & 2668-140.17=16 & (\text{a-156b}) & -156 \\
 \text{(d)} & 17-16=1 & (\text{b-c}) & 157
 \end{array}$$

La valeur de d est donc 157.

Malgré la relative simplicité des fonctions D et E, elles ne sont pas aussi faciles à calculer que leurs équivalents pour le DES : l'exponentiation *mod p* est une opération plus coûteuse en temps que des permutations ou des recherches dans une table lorsque p est grand. Cela rend plus compliqué la conception d'un composant électronique destiné à chiffrer et déchiffrer rapidement en utilisant cette approche.

4.3 Cryptographie à clé publique

La notion de cryptographie à clé publique a été introduite par Diffie et Hellman en 1976. Les systèmes à clé publique, également appelés double-clé

ou asymétrique, différent des précédents en ce sens qu'il n'y a plus une seule clé partagée par deux utilisateurs. À la place chaque utilisateur a sa propre clé qui est divisée en deux parties : une privée et une publique. À partir de la partie publique on peut générer une transformation E , à partir de la partie privée une transformation D . Par analogie avec les systèmes vus jusqu'à présent, on peut appeler les fonctions E et D fonction de chiffrement et de déchiffrement mais nous verrons qu'un système peut satisfaire $D(E(M)) = M$ ou $E(D(M)) = M$, voire les deux.

4.3.1 Secret et authentification

Pour assurer le secret, les transformations d'un système à clé publique doivent satisfaire $D(E(M)) = M$. Supposons que A veuille envoyer un message secret M à B . A doit alors avoir accès à E_B , la fonction de transformation publique de B . A va chiffrer M , $C = E_B(M)$ et envoyer le résultat C à B . À la réception B va utiliser sa fonction privée de transformation D_B pour déchiffrer. Si la transmission de A est espionnée l'intrus ne pourra déchiffrer C car la fonction D_B est secrète donc le secret est assuré. Par contre, comme E_B est publique, B n'a aucun moyen de connaître l'identité de l'envoyeur. De même le message envoyé par A peut être altéré. Les propriétés d'authentification et d'intégrité ne sont donc pas assurées.

Pour qu'elles le soient les transformations doivent satisfaire la propriété $E(D(M)) = M$. En effet, supposons que A veuille envoyer un message authentifié M à B . Cela signifie que B doit pouvoir vérifier que le message a bien été envoyé par A et qu'il n'a pas été altéré en chemin. Pour cela A va utiliser sa transformation privée D_A , calculer $C = D_A(M)$ et envoyer C à B . B peut alors utiliser la transformation publique E_A pour calculer $E_A(C) = E_A(D_A(M)) = M$. En supposant que M représente un texte "valide" (au sens du protocole utilisé), B est sûr que le message a été envoyé par A et n'a pas subi d'altérations pendant le transport. Cela vient de la nature uni-directionnelle de E_A : si un attaquant pouvait, à partir d'un message M , trouver C' tel que $E_A(C') = M$ cela signifierait qu'il peut calculer l'inverse de la fonction E_A ce qui est une contradiction. Par contre dans ce cas le secret n'est pas assuré car tout le monde peut accéder à E_A et donc déchiffrer le message.

Exercice

En utilisant les fonctions E_A , D_A , E_B et D_B trouver un couple de fonctions de chiffrement et déchiffrement satisfaisant à la fois le secret et l'authentification.

Solution : Pour garantir le secret et l'authentification il faut qu'un domaine commun existe entre A et B , ou plus précisément entre E_A , D_A , E_B et D_B . A envoie alors $C = E_B(D_A(M))$ à B ; B calcule alors $E_A(D_B(C)) = E_A(D_A(M))$. Un espion ne peut pas décoder C car il ne dispose pas de D_B et le secret est donc préservé. Si un espion envoie un message C' à la place

de C , C' ne pourra pas être décodé en un message valide M car pour cela il faudrait connaître D_A ce qui assure l'authenticité.

L'algorithme utilisé principalement pour la cryptographie à clé publique est RSA. Il satisfait notamment les propriétés

Propriété 9 $E(D(M)) = M$

et

Propriété 10 $D(E(M)) = M$

4.3.2 L'algorithme RSA

Dans le système RSA un utilisateur crée son couple (clé publique, clé privée) en utilisant la procédure suivante :

1. Choisir au hasard deux grands nombres premiers p et q . Il faut que p et q contiennent au moins 100 chiffres décimaux chacun.
2. Calculer $n = pq$
3. Choisir un petit entier e qui est premier avec $\phi(n) = (p-1)(q-1)$
4. Calculer d , l'inverse par la multiplication de e , modulo $\phi(n)$. Le théorème 16 garantit que e existe et est unique.
5. Publier la paire $P = (e, n)$ comme sa clé publique RSA.
6. Garder secrète la paire $S = (d, n)$ qui est sa clé privée RSA.

Les opérations de transformation sont alors

Définition 16 $E_A(M) = M^e \text{ mod } n$

et

Définition 17 $D_A(C) = C^d \text{ mod } n$

Théorème 17 Les transformations E et D satisfont les propriétés 9 et 10 pour $M \in \mathbf{Z}_n$.

Preuve

On commencera par rappeler le théorème de Fermat :

Théorème 18 Si p est premier alors $a^{p-1} \equiv 1 \pmod{p}$ pour tout $a \in \mathbf{Z}_n^*$

Ce théorème se démontre en utilisant le théorème d'Euler qui nous dit que pour tout entier $n > 1$ et $a \in \mathbf{Z}_n^*$ on a : $a^{\phi(n)} \equiv 1 \pmod{n}$ avec $\phi(n)$ cardinal de \mathbf{Z}_n^* et donc $\phi(p) = p-1$ si p premier.

Des deux définitions 16 et 17, on déduit que pour tout $M \in \mathbf{Z}_n$ $E(D(M)) = D(E(M)) = M^{ed} \text{ mod } n$. Comme e et d sont des inverses pour la multiplication modulo $\phi(n) = (p-1)(q-1)$, on a :

$ed = 1 + k(p-1)(q-1)$ pour un entier k . Mais, si $M \not\equiv 0 \pmod{p}$ on a (théorème de Fermat) :

$$M^{ed} \equiv M(M^{p-1})^{k(q-1)} \pmod{p} \quad (4.5)$$

$$\equiv M(1)^{k(q-1)} \pmod{p} \quad (4.6)$$

$$\equiv M \pmod{p} \quad (4.7)$$

Comme on a plus, $M^{ed} \equiv M \pmod{p}$ si $M \equiv 0 \pmod{p}$ donc

$M^{ed} \equiv M \pmod{p}$ pour tout M . De même,

$M^{ed} \equiv M \pmod{q}$ pour tout M . Ce qui donne, en utilisant un corollaire du théorème Chinois des restes³,

$M^{ed} \equiv M \pmod{n}$. pour tout M . cqfd.

Exercice

Calculer une clé publique et une clé privée pour $p = 47$ et $q = 59$. Pour simplifier la chose on prendra $e = 17$. Chiffrer le message 0003 avec la clé publique et vérifier que la clé privée permet bien de retrouver le message initial.

– Prenons $p = 47$ et $q = 59$ ⁴.

– on calcule $n = p * q = 47 * 59 = 2773$

– on choisit e , premier par rapport à n . Prenons par exemple $e = 17$.

– On calcule alors, par l'algorithme d'Euclide étendu, d tel que $d.e = 1 \pmod{(p-1).(q-1)}$, soit $d = 157$

On a alors un couple clé publique (17,2773) clé privée (157,2773).

Si on veut chiffrer la valeur '0003' on calculera $(0003^{17}) \pmod{2773}$ c'est-à-dire 1553. Pour retrouver le message d'origine on calculera $(1553^{157}) \pmod{2773}$ qui donne bien 3.

La sécurité fournie par RSA repose essentiellement sur la difficulté à factoriser de grands entiers. Si un attaquant peut factoriser le modulo n dans une clé publique, il peut alors déduire directement la clé privée de la clé publique. Donc si la factorisation de grands entiers devient facile, casser RSA devient facile aussi. La réciproque, si factoriser de grands entiers est dur alors casser RSA est dur, n'a pas été prouvée. Cependant, après une décade de recherche, aucun moyen plus efficace que la factorisation de n n'a été découvert pour casser RSA.

³Si n_1, n_2, \dots, n_k sont premiers entre eux et que $n = n_1 n_2 \dots n_k$ alors pour tous entiers a et x

$$x \equiv a \pmod{n_i}$$

pour $i = 1, 2, \dots, k$ si et seulement si

$$x \equiv a \pmod{n}$$

⁴Ces valeurs sont faibles et ne correspondent évidemment pas à des clé réelles.

4.3.3 Utilisation de RSA

D'un point de vue performances, un algorithme de type RSA est environ 10,000 fois plus lent qu'un DES. Pour des clés de 512 bits l'ordre de grandeur de capacité de chiffrement avec la technologie actuelle est de 64Kbits/s, les 1Mbits/s sont envisageables à court ou moyen terme.

La solution pour un chiffrement efficace réside dans l'utilisation conjointe de systèmes de chiffrement symétrique et asymétrique. Par exemple PGP⁵ chiffre un message avec le protocole suivant :

1. Le texte source est chiffré avec une clé K de type IDEA ou DES
2. La clé K est chiffrée selon le principe RSA avec la clé publique du destinataire
3. Envoi du message composé de la clé K chiffrée par RSA et d'un texte chiffré selon un algorithme IDEA ou DES

Le récepteur effectuera alors les opérations de déchiffrement suivantes :

1. Déchiffrement de la clé grâce à la clé privée D en utilisant RSA
2. Déchiffrement du texte en utilisant DES ou IDEA avec la clé K

Pour l'authentification le principe est un peu différent :

- Génération d'un code qui identifie, par un nombre de taille fixe, le texte
- Chiffrement de ce code en utilisant RSA avec la clé privée de l'émetteur du message

Cela permet au correspondant de vérifier la validité du message en utilisant la clé publique de l'expéditeur.

⁵PGP (Pretty Good Privacy) est l'outil le plus utilisé pour le chiffrement de messages ou l'authentification par courrier électronique.