Université Bordeaux 1
Licence de Sciences, Technologies, Santé
Mentions Mathématiques et Informatique
M1MI2016 Codes et Cryptologie

Codes et Cryptologie

Christine Bachoc

Introduction

L'objectif de ce cours, qui est un cours optionnel de première année, deuxième semestre de Licence de Sciences, Technologies, Santé, Mention Mathématiques et Informatique de l'Université Bordeaux 1, proposé depuis l'année universitaire 2011/2012, est de présenter aux étudiants quelques applications au traitement de l'information des mathématiques qui leurs sont enseignées.

On abordera dans ce cours deux domaines liées à la protection de l'information :

La cryptologie étudie les problématiques liées à la *sécurité* des échanges d'information, en particulier cherche des solutions aux problèmes de confidentialité et d'authentification. Le célèbre trio Alice, Bob et Oscar joue un scénario vieux comme le monde : Alice et Bob veulent échanger un message m à l'insu d'Oscar, qui, lui, va faire tout ce qui est en son pouvoir pour en prendre connaissance. Il peut aussi tenter d'autres actions déplaisantes comme de communiquer avec Alice en prétendant être Bob, ou bien de remplacer, à leur insu bien sûr, m par un message de son choix.

La théorie des codes correcteurs d'erreurs répond au problème de la *fiabilité*, en cherchant à protéger l'information d'une détérioration due à la présence de bruit. La solution consiste à apporter de la redondance à l'information transmise, afin que celle-ci puisse être restaurée - complètement ou du moins autant que possible - même si elle a été perturbée.

Le cours s'attachera à présenter une perspective historique de ces domaines. La cryptologie a une longue histoire derrière elle, remontant à l'antiquité, et riche en enseignements sur les principes de mise en oeuvre des méthodes modernes. La correction d'erreurs s'est développée plus récemment, accompagnant les progrès de l'informatique et des télécommunications. On y verra les rôles joués par des personnages célèbres de l'histoire des sciences comme Charles Babbage, Alan Turing, Claude Shannon.

Dans la mesure du possible, compte tenu du corpus des connaissances en première année, nous étudierons les développements actuels de ces domaines, qui sont devenus indispensables aux technologies modernes (carte bancaire, téléphonie mobile, disques compacts, etc..).

Les outils mathématiques mis en oeuvre seront essentiellement de l'arithmétique en cryptologie, et de l'algèbre linéaire (sur un corps à deux éléments !) en théorie des codes. Bien sûr, ce cours sera l'occasion de consolider et d'étoffer les connaissances des étudiants sur ces sujets.

Il n'est pas question dans un cours de ce niveau d'être exhaustifs sur aucun des domaines abordés. Notre objectif sera atteint si nous arrivons à convaincre les étudiants de l'intérêt et de la puissance des outils mathématiques qui leur sont enseignés. Nous espérons que leur curiosité sera suffisamment stimulée pour qu'ils souhaitent aller plus loin dans l'approfondissement de leur apprentissage mathématique d'une part et de leur connaissance de ces domaines passionnants d'autre part.

Table des matières

Chapitre 1

Arithmétique

Notations : l'ensemble des entiers relatifs est noté $\mathbb{Z}$. Celui des entiers naturels est noté $\mathbb{N}$. L'expression "un entier" désigne toujours un entier relatif.

Les résultats énoncés dans ce chapitre sont pour partie des rappels de l'UE *Fondamentaux pour les Mathématiques et l'Informatique* du premier semestre.

1.1 Division euclidienne

Définition 1.1.1. Soient a et b deux éléments de $\mathbb{Z}$. Si b est non nul, on dit que a est *divisible* par b ou que a est un *multiple* de b ou encore que b *divise* a, s'il existe un élément q dans $\mathbb{Z}$ tel que $a = bq$. On notera alors $b|a$, et on lit b divise a.

Proposition 1.1.2. On a les propriétés suivantes : soient a, b et c trois éléments de $\mathbb{Z}$ non nuls,

1. si c divise a et b, il divise toute expression de la forme $ua + vb$, où $u \in \mathbb{Z}$ et $v \in \mathbb{Z}$, en particulier il divise $a + b$ et $a - b$;
2. si $c|b$ et $b|a$ alors $c|a$;
3. si $a|b$ et $b|a$ alors $a = \pm b$;
4. $1|a$ et $a|a$;
5. $a|0$.

Définition 1.1.3. On dit qu'un entier naturel $n > 1$ est *premier* si ses seuls diviseurs positifs sont 1 et lui-même.

Comme le montre le théorème suivant, les nombres premiers sont les 'briques élémentaires' qui permettent de fabriquer tous les entiers par multiplication. Il est d'une grande importance théorique, mais il faut savoir que la factorisation effective d'un entier en produit de nombres premiers est un problème algorithmiquement difficile. C'est précisément cette difficulté qui est exploitée dans certains systèmes cryptographiques dont le système RSA qui sera discuté plus tard.

Théorème 1.1.4. Tout entier $n > 1$ s'écrit de manière unique

$$n = p_1^{k_1} p_2^{k_2} \dots p_s^{k_s}$$

où $\begin{cases} \text{les entiers } p_i \text{ sont premiers et vérifient } p_1 < p_2 < \dots < p_s \\ \text{les entiers } k_i \text{ sont strictement positifs} \end{cases}$

Théorème 1.1.5. Pour tout couple d'entiers (a, b) avec $b \neq 0$, il existe un unique couple (q, r) d'entiers tels que

$$a = bq + r \quad \text{avec } 0 \leq r < |b|.$$

Les entiers q et r sont respectivement le *quotient* et le *reste* de la division euclidienne de a par b.

1.2 pgcd, ppcm et théorème de Bézout

Définition 1.2.1. Soient a et b deux entiers non tous les deux nuls. On appelle *plus grand commun diviseur*, le plus grand entier positif qui divise à la fois a et b. On le note :

$$\text{pgcd}(a, b).$$

Définition 1.2.2. On dit que deux entiers a et b sont *premiers entre eux* si leur pgcd est égal à 1.

Remarque 1.2.3. Si $d = \text{pgcd}(a, b)$, alors il existe des entiers a' et b' tels que $a = da'$ et $b = db'$. De plus, a' et b' sont premiers entre eux. En effet, s'ils avaient un diviseur commun $c > 1$, alors dc serait un diviseur commun de a et b plus grand que d.

Définition 1.2.4. Soient a et b deux entiers non tous les deux nuls. On appelle *plus petit commun multiple*, le plus petit entier positif qui soit multiple de a et b. On le note :

$$\text{ppcm}(a, b).$$

Proposition 1.2.5. Soient a et b deux entiers non tous les deux nuls et m leur ppcm. Alors, si n est un multiple commun à a et b, il est aussi multiple de m.

Démonstration. On effectue la division euclidienne de n par m : $n = mq + r$, $0 \leq r < m$. Alors $a|n$ et $a|m$ donc $a|r$; de même $b|r$. Ainsi, r, s'il est non nul, fournit un multiple commun à a et b strictement plus petit que m, une contradiction. □

Théorème 1.2.6. (Théorème de Bézout) Soient a et b deux entiers non tous les deux nuls. Si $d = \text{pgcd}(a, b)$ alors il existe deux entiers u et v tels que

$$d = au + bv.$$

Démonstration. On pose $D := \{au + bv : u, v \in \mathbb{Z}\}$, et $d = \text{pgcd}(a, b)$. Soit d' le plus petit entier positif non nul contenu dans D. On va montrer que $d' = d$.

Monrtrons d'abord que d divise d'. En effet, comme $d' \in D$, il existe $u, v \in \mathbb{Z}$ tels que $d' = au + bv$. Donc, comme $d|a$ et $d|b$, d divise $au + bv = d'$.

Montrons ensuite que d' divise a et b. En effet, si on effectue la division euclidienne de a par d', $a = d'q + r$, avec $0 \leq r < d'$, on voit facilement que $a - d'q$ est dans D, donc r aussi. Comme $0 \leq r < d'$, et que d' est le plus petit élément positif de D, on en déduit que $r = 0$, et donc que d' divise a. Par le même raisonnement, d' divise b.

Comme d' divise a et b et $d = \text{pgcd}(a, b)$, on obtient que $d' \leq d$. Mais comme on a montré que d divise d', on peut conclure que $d = d'$. En particulier, $d \in D$ donc il existe bien des entiers u, v tels que $d = au + bv$. □

Remarque 1.2.7. La relation dans le théorème ci-dessus est appelée *une identité de Bézout*. Attention, elle n'est pas unique : en fait il existe plusieurs choix pour les entiers u et v.

Remarque 1.2.8. Le théorème précédent est équivalent à la propriété : si $d = \text{pgcd}(a, b)$, et si $a\mathbb{Z} + b\mathbb{Z} := \{au + bv : u, v \in \mathbb{Z}\}$,

$$d\mathbb{Z} = a\mathbb{Z} + b\mathbb{Z}.$$

1.3 Algorithme d'Euclide étendu

C'est un algorithme qui permet de calculer $d = \text{pgcd}(a, b)$, ainsi que deux entiers u et v tels que $au + bv = d$, c'est-à-dire une relation de Bézout, *de façon simultanée*. On suppose $a \geq b > 0$.

On calcule une suite $r_0, r_1, \ldots, r_k, \ldots$ de restes obtenus par divisions euclidiennes successives à partir de $r_0 = a$, $r_1 = b$,

$$\begin{cases} r_0 = r_1 q_1 + r_2 & \text{où} \quad 0 \leq r_2 < r_1 \\ r_1 = r_2 q_2 + r_3 & \text{où} \quad 0 \leq r_3 < r_2 \\ \quad \ldots & \ldots \quad \ldots \\ r_{k-2} = r_{k-1} q_{k-1} + r_k & \text{où} \quad 0 \leq r_k < r_{k-1} \\ r_{k-1} = r_k q_k + r_{k+1} & \text{où} \quad 0 \leq r_{k+1} < r_k \end{cases}$$

ainsi que deux autres suites d'entiers u_k et v_k définis récursivement de la façon suivante : on pose $u_0 = 1$, $v_0 = 0$, $u_1 = 0$, $v_1 = 1$ et pour $k \geq 1$

$$\begin{cases} u_{k+1} = u_{k-1} - u_k q_k \\ v_{k+1} = v_{k-1} - v_k q_k \end{cases}$$

On a alors :

Proposition 1.3.1. Pout tout $k \geq 0$,

1. $\text{pgcd}(a, b) = \text{pgcd}(r_k, r_{k+1})$.
2. $r_k = au_k + bv_k$

Démonstration. Par récurrence sur k. La première propriété découle du lemme suivant :

Lemme 1.3.2. Soient a et b deux entiers, avec $b \neq 0$. Si r est le reste de la division euclidienne de a par b alors

$$\text{pgcd}(a, b) = \text{pgcd}(b, r).$$

Démonstration. (du Lemme). Si $a = bq + r$, alors $d|a$ et $d|b$, si et seulement si $d|b$ et $d|r$. □

La deuxième est bien vraie pour $k = 0$, puisque $r_0 = a$, $u_0 = 1$, $v_0 = 0$. On calcule par récurrence $au_{k+1} + bv_{k+1}$:

$$\begin{aligned} au_{k+1} + bv_{k+1} &= a(u_{k-1} - u_k q_k) + b(v_{k-1} - v_k q_k) \\ &= (au_{k-1} + bv_{k-1}) - (au_k + bv_k)q_k \\ &= r_{k-1} - r_k q_k \quad \text{(par l'hypothèse de récurrence)} \\ &= r_{k+1}. \end{aligned}$$

□

La suite des restes $(r_0, r_1, r_2, \dots)$ étant une suite strictement décroissante d'entiers positifs, on obtient nécessairement un reste nul au bout d'un nombre fini de divisions. Soit r_n le dernier reste non nul. On a donc $r_{n+1} = 0$ et par conséquent :

$$\text{pgcd}(a, b) = \text{pgcd}(r_n, r_{n+1}) = \text{pgcd}(r_n, 0) = r_n = au_n + bv_n.$$

Donc

$$d = r_n \quad \text{et} \quad d = au_n + bv_n.$$

Dans la pratique, on calcule simultanément les nombres r_k, q_k, u_k, v_k que l'on peut disposer dans un tableau, jusqu'à obtenir un reste nul.

Exemple : $a = 366$, $b = 56$.

r_k	u_k	v_k	q_k	
366	1	0		
56	0	1	6	$(366 = 6 * 56 + 30)$
30	1	-6	1	$(56 = 1 * 30 + 26)$
26	-1	7	1	$(30 = 1 * 26 + 4)$
4	2	-13	6	$(26 = 6 * 4 + 2)$
2	-13	85	2	$(4 = 2 * 2 + 0)$
0				

On a trouvé : $\text{pgcd}(366, 56) = 2$ et $2 = 366 * -13 + 56 * 85$.

1.4 Quelques conséquences du théorème de Bézout

Proposition 1.4.1. Soient a, b deux entiers, d leur pgcd. Si d' est un diviseur commun de a et b, alors d' divise d.

Démonstration. C'est immédiat en écrivant une relation de Bézout $d = au + bv$. □

Remarque : Le pgcd de a et b, défini initialement comme plus grand élément, *au sens de la relation d'ordre usuelle*, de l'ensemble des diviseurs communs à a et b est donc également le plus grand élément de ce même ensemble, *au sens de la divisibilité.*

Proposition 1.4.2. Soit a et b des entiers. Alors $\text{pgcd}(a, b) = 1$ si et seulement s'il existe deux entiers u et v tels que $au + bv = 1$.

Démonstration. Si $\text{pgcd}(a, b) = 1$, le théorème de Bézout 1.2.6 montre qu'il existe u et v tels que $1 = au+bv$. Réciproquement, si on a une relation de la forme $1 = au+bv$, alors un diviseur commun à a et b, divise $au + bv$ donc divise 1 donc vaut ± 1. On a bien montré que a et b sont premiers entre eux. □

Proposition 1.4.3. Soient a et b deux entiers dont l'un au moins est non nul. Alors

$$\text{ppcm}(a, b)\,\text{pgcd}(a, b) = |ab|.$$

Démonstration. On peut supposer sans perte de généralité que a et b sont positifs. Soit $d = \text{pgcd}(a, b)$ et $m = \text{ppcm}(a, b)$. On pose $a = da'$, $b = db'$, avec a' et b' deux entiers premiers

entre eux. Comme $d|a$, on a aussi $d|ab$, donc il existe un entier m' tel que $dm' = ab$. On veut montrer que $m' = m$.

En remplaçant $a = da'$ et $b = db'$ dans $dm' = ab$ on obtient que $m' = da'b' = ab' = a'b$ donc m' est un multiple commun de a et b donc en appliquant la proposition 1.2.5 m' est un multiple de m. D'autre part, puisque m est un multiple commun de a et b, il existe des entiers q et r tels que $m = aq = br$. Alors $ma' = rba' = rm'$ et $mb' = qab' = qm'$ donc m' divise ma' et mb'. Comme a' et b' sont premiers entre eux, il existe une relation de Bézout $1 = a'u + b'v$; donc m' divise $ma'u + mb'v = m$. Finalement, $m|m'$ et $m'|m$ donc $m = m'$. □

Théorème 1.4.4. *(Lemme de Gauss).* Soient a, b et c trois entiers. Si a divise bc et si a est premier avec b, alors a divise c.

Démonstration. Puisque $\text{pgcd}(a, b) = 1$, par le théorème 1.2.6, il existe u et v tels que $1 = au+bv$. En multipliant par c, on obtient $c = acu+bcv$. Comme $a|bc$, $a|(acu+bcv)$ donc $a|c$. □

Proposition 1.4.5. Soient a, b et c trois entiers.

1. Si a divise c et b divise c et si $\text{pgcd}(a, b) = 1$ alors ab divise c.
2. Si $\text{pgcd}(a, b) = 1$ et si $\text{pgcd}(a, c) = 1$ alors $\text{pgcd}(a, bc) = 1$.
3. Si p est un nombre premier et si p divise ab alors p divise a ou p divise b.

Démonstration. 1. Si $a|c$ et $b|c$ alors $\text{ppcm}(a, b)|c$. Mais, si $\text{pgcd}(a, b) = 1$, $ab = \text{ppcm}(a, b)$.

2. Si $d|a$ et $d|bc$, comme $\text{pgcd}(a, b) = 1$, on aura aussi $\text{pgcd}(d, b) = 1$. Par le lemme de Gauss, $d|c$. On a : $d|a$ et $d|c$, donc $d|\,\text{pgcd}(a, c) = 1$ donc $d = 1$.

3. Pour un nombre premier p, et a quelconque, on a soit $p|a$ soit $\text{pgcd}(a, p) = 1$. Donc, si p ne divise pas a, on peut appliquer le lemme de Gauss et en déduire que $p|b$. □

1.5 La factorisation en produit de nombres premiers

Dans cette section on traduit certaines des notions vues précédemment en termes de la factorisation d'un entier en produit de nombres premiers. Dans ce but il est plus agréable de l'écrire

$$a = \pm \prod_{p_i \in \mathcal{P}} p_i^{k_i} \quad \text{avec } k_i \geq 0$$

où le produit porte sur l'ensemble (infini) $\mathcal{P}$ de tous les nombres premiers, mais où seul un nombre fini des exposants k_i sont non nuls.

Proposition 1.5.1. Avec les notations précédentes, soit $a = \pm \prod_{p_i \in \mathcal{P}} p_i^{k_i}$, $k_i \geq 0$ et soit $b = \pm \prod_{p_i \in \mathcal{P}} p_i^{\ell_i}$, $\ell_i \geq 0$.

1. a divise b si et seulement si $k_i \leq \ell_i$ pour tout i.
2. $\text{pgcd}(a, b) = \prod_i p_i^{\min(k_i, \ell_i)}$.
3. $\text{ppcm}(a, b) = \prod_i p_i^{\max(k_i, \ell_i)}$.

Démonstration. Laissée en exercice. □

Remarque 1.5.2. Pour calculer le pgcd de deux entiers grands (et même de quelques chiffres décimaux) il est plus efficace d'appliquer l'algorithme d'Euclide étendu plutôt que de passer par leur décomposition en produit de nombres premiers.

Chapitre 2

L'anneau $\mathbb{Z}/n\mathbb{Z}$

2.1 Entiers modulo n

Définition 2.1.1. Soit $n > 1$. On dit que deux entiers a et b sont *congrus modulo* n si $a - b$ est un multiple de n, autrement dit s'il existe un entier q tel que $a - b = qn$.

Les expressions : être dans la même classe résiduelle modulo n, ou être dans la même classe de congruence modulo n, ou plus simplement être égaux modulo n, sont synonymes d'être congrus modulo n.

On utilise indifféremment les notations :

$$a \equiv b \bmod n,\ a \equiv b\ (n),\ a = b \bmod n,\ a = b\ (n).$$

Proposition 2.1.2. La relation "$a \equiv b \bmod n$" est une relation d'équivalence sur $\mathbb{Z}$. Les classes d'équivalence sont appelées les *classes de congruence modulo* n. Leur ensemble est noté $\mathbb{Z}/n\mathbb{Z}$.

Démonstration. On vérifie que cette relation est rêflexive, symmétrique, et transitive. □

Les expressions : classes de congruence modulo n, classes résiduelles modulo n, ou simplement classes modulo n sont synonymes.

La classe de a modulo n est notée "$a \bmod n$". On dit que a est un *représentant* de sa classe. Une classe donnée a plusieurs représentants, c'est-à-dire on peut avoir $a \bmod n = b \bmod n$, précisément lorsque $a - b$ est un multiple de n.

Proposition 2.1.3. Les classes de a et b modulo n sont égales, i.e. $a = b \bmod n$, si et seulement si a et b ont le même reste dans la division euclidienne par n.

Démonstration. Supposons que $a = b \bmod n$. Alors $a - b$ est un multiple de n, donc $a = b + cn$. Effectuons la division euclidienne de b par n : $b = nq + r$ avec $0 \leq r < n$. On obtient $a = n(q + c) + r$. Donc le reste de la division de a par n est bien r, le même que celui de b.

Réciproquement, supposons que a et b aient le même reste dans la division par n. Alors $a = nq + r$ et $b = nq' + r$. Alors $a - b = n(q - q')$ est un multiple de n donc $a = b \bmod n$. □

Les restes possibles dans la division euclidienne par n sont : $0, 1, \ldots, (n-1)$. Il y a donc exactement n classes distinctes modulo n et

$$\mathbb{Z}/n\mathbb{Z} = \{0, 1, 2, \ldots, (n-1) \bmod n\}.$$

2.2 Addition et multiplication dans $\mathbb{Z}/n\mathbb{Z}$

Théorème 2.2.1. L'addition et la multiplication sont compatibles avec la relation de congruence modulo n, c'est-à-dire :

$$\text{si } \begin{cases} a_1 = b_1 \bmod n \\ a_2 = b_2 \bmod n \end{cases}, \text{ alors } \begin{cases} a_1 + a_2 = b_1 + b_2 \bmod n \\ a_1 a_2 = b_1 b_2 \bmod n. \end{cases}$$

Démonstration. On traduit les propriétés de congruences supposées : il existe q_1 tel que $a_1 = b_1 + q_1 n$ et q_2 tel que $a_2 = b_2 + q_2 n$. Alors,

$$\begin{aligned} a_1 + a_2 &= b_1 + q_1 n + b_2 + q_2 n \\ &= (b_1 + b_2) + (q_1 + q_2) n \end{aligned}$$

donc $a_1 + a_2 = b_1 + b_2 \bmod n$. De même,

$$\begin{aligned} a_1 a_2 &= (b_1 + q_1 n)(b_2 + q_2 n) \\ &= b_1 b_2 + (b_1 q_2 + q_1 b_2 + q_1 q_2 n) n \\ &= b_1 b_2 \bmod n. \end{aligned}$$

□

Corollaire 2.2.2. On définit l'addition et la multiplication dans $\mathbb{Z}/n\mathbb{Z}$ par :

$$\begin{cases} (a \bmod n) + (b \bmod n) = (a + b) \bmod n \\ (a \bmod n)(b \bmod n) = ab \bmod n. \end{cases}$$

Démonstration. le théorème 2.2.1 montre précisément que ces définitions ont bien un sens, puisque le résultat de l'addition ou de la multiplication de deux classes ne dépend pas du représentant choisi. □

Pour alléger les notations, quand on calcule modulo n, on écrit une seule fois par ligne l'expression "mod$n$" ou "(n)". Exemples :

$$\begin{aligned} &4^2 + 4 + 1 = 16 + 4 + 1 = 1 + 4 + 1 = 6 = 1 \bmod 5 \\ &(x+1)^3 = x^3 + 3x^2 + 3x + 1 = x^3 + 1 \bmod 3. \end{aligned}$$

Les rêgles de calcul usuelles dans $\mathbb{Z}$ restent valables dans $\mathbb{Z}/n\mathbb{Z}$. En particulier, on a :
- $x + 0 = x \bmod n$
- $x + (-x) = 0 \bmod n$ (*l'opposé* de $x \bmod n$ est $-x \bmod n$)
- $x.1 = x \bmod n$, $x.0 = 0 \bmod n$
- $x(y + z) = xy + xz \bmod n$
- $(x \bmod n)^k = x^k \bmod n$, etc...

On dit que $(\mathbb{Z}/n\mathbb{Z}, +, \times)$ est un *anneau commutatif et unitaire.*

2.3 Inversibles de $\mathbb{Z}/n\mathbb{Z}$

Définition 2.3.1. Soit a un entier, on dit que a est *inversible modulo* n ou que $a \bmod n$ est inversible dans $\mathbb{Z}/n\mathbb{Z}$, s'il existe b tel que $ab = 1 \bmod n$. L'ensemble des inversibles de $\mathbb{Z}/n\mathbb{Z}$ est noté $(\mathbb{Z}/n\mathbb{Z})^*$. On note $a^{-1} \bmod n$ l'inverse de $a \bmod n$.

Exemple 2.3.2. Les inversibles de $\mathbb{Z}/6\mathbb{Z}$ sont :

$$(\mathbb{Z}/6\mathbb{Z})^* = \{1, 5 \bmod 6\}.$$

Proposition 2.3.3. Si a est inversible modulo n, son inverse est unique modulo n.

Démonstration. En effet, si $ab = 1\ (n)$ et si $ac = 1\ (n)$, en multipliant cette dernière égalité par b, on trouve $abc = b\ (n)$ soit $(ab)c = b\ (n)$ mais comme $ab = 1\ (n)$, on obtient $c = b\ (n)$. □

Remarque 2.3.4. 0 n'est jamais inversible modulo n puisque $0.b = 0 \bmod n$ et 1 est toujours inversible puisque $1.1 = 1 \bmod n$.

Théorème 2.3.5. a est inversible modulo n si et seulement si $\text{pgcd}(a, n) = 1$. Dans ce cas, l'inverse de a modulo n est donné par une relation de Bezout entre a et n : si $au + nv = 1$, alors $a^{-1} = u \bmod n$.

Démonstration. Supposons a inversible modulo n, alors il existe b tel que $ab = 1 \bmod n$. Cela signifie qu'il existe q tel que $ab = 1 + qn$, soit $ab - qn = 1$. Par le corollaire 1.4.2, on en déduit que a et n sont premiers entre eux.

Réciproquement, supposons que a et n soient premiers entre eux. Alors par le théorème de Bézout, il existe u et v tels que $au + nv = 1$, ce qui implique $au = 1 \bmod n$. Donc a est inversible modulo n d'inverse $u \bmod n$. □

En résumé, nous avons démontré que

$$(\mathbb{Z}/n\mathbb{Z})^* = \{a \bmod n : 1 \leq a \leq n-1 \text{ et } \text{pgcd}(a, n) = 1\}.$$

La bonne méthode pour déterminer si a est inversible modulo n et pour calculer son inverse s'il existe, consiste à effectuer l'algorithme d'Euclide étendu.

Le cas où n est un nombre premier est particulièrement intéressant. En effet, dans ce cas, tout élément non nul de $\mathbb{Z}/n\mathbb{Z}$ est inversible.

Proposition et définition 2.3.6.

1. Si p est un nombre premier, tout élément non nul de $\mathbb{Z}/p\mathbb{Z}$ est inversible.
2. Réciproquement, si n est un entier, tel que tout élément non nul de $\mathbb{Z}/n\mathbb{Z}$ est inversible, alors n est premier.
3. Un *diviseur de zéro* de $\mathbb{Z}/n\mathbb{Z}$ est un entier a tel que $a \neq 0 \bmod n$ et tel qu'il existe $b \neq 0 \bmod n$ tel que $ab = 0 \bmod n$. Si a est un diviseur de zéro modulo n, alors il ne peut pas être inversible modulo n.

Démonstration. 1. Soit $1 \leq a \leq p-1$, et $d > 0$ un diviseur commun de a et de p. Comme p est premier, $d = 1$ ou $d = p$. Comme a est plus petit que p, la seule possibilité est $d = 1$. Donc $\text{pgcd}(a, p) = 1$ et a est inversible modulo p.

2. Si n n'était pas premier, il aurait un diviseur $d > 0$ différent de 1 et n. Alors $d \neq 0 \bmod n$ et $\text{pgcd}(d, n) = d > 1$ donc d est non nul et n'est pas inversible modulo n, ce qui est contraire à l'hypothèse.

3. En effet, si a est inversible, il existe c tel que $ac = 1 \bmod n$. En multipliant $ab = 0 \bmod n$ par c on obtient $b = 0 \bmod n$. Donc a ne peut pas être un diviseur de zéro modulo n. □

2.4 Application : résoudre une équation du premier degré modulo n

Pour faire des calculs dans $\mathbb{Z}/n\mathbb{Z}$, le principe général est simple : on peut tout faire comme si on avait des nombres réels ou rationnels, sauf diviser, que l'on peut remplacer par la multiplication par l'inverse, s'il existe. Voici quelques exemples :

On veut résoudre l'équation $2x-3=0 \bmod 5$. On commence à isoler x : $2x=3 \bmod 5$. On a envie de diviser par 2, que l'on remplace par la multiplication par l'inverse de 2. Justement, 2 est inversible modulo 5 puisque premier à 5 ; $2.3=1 \bmod 5$, son inverse est 3. On multiplie par 3 l'équation $2x=3 \bmod 5$ ce qui conduit à $x=9=-1 \bmod 5$. Donc $x=-1 \bmod 5$ est l'unique solution.

Si on était partis de $2x-3=0 \bmod 10$, soit $2x=3 \bmod 10$, on n'aurait pas pu inverser 2 modulo 10. Mais cette équation ne peut pas avoir de solution car 2 ne divise pas 3.

Dernier cas possible : $2x-2=0 \bmod 10$ soit $2x=2 \bmod 10$ mais dans ce cas on peut diviser par 2 tous les termes (y compris le module) dans $\mathbb{Z}$ et cette équation équivaut à $x=1 \bmod 5$. Il faut remarquer que l'équation initiale a deux solution modulo 10 qui sont 1 et 6.

Dans le cas général, on a :

Proposition 2.4.1. Pour résoudre l'équation $ax+b=0 \bmod n$, on est dans l'un des cas suivants :

1. Si a est inversible modulo n, $x=-a^{-1}b \bmod n$ est l'unique solution.
2. Si $d=\mathrm{pgcd}(a,n)>1$, et d ne divise pas b, il n'y a pas de solutions.
3. Si $d=\mathrm{pgcd}(a,n)>1$, et $d|b$, on pose $a=da_1$, $b=db_1$, $n=dn_1$, avec $\mathrm{pgcd}(a_1,n_1)=1$. L'équation $ax+b=0 \bmod n$ est équivalente à l'équation $a_1x+b_1=0 \bmod n_1$ qui se résoud comme en 1. et a une solution unique $x=-a_1^{-1}b_1 \bmod n_1$. Soit $k:=-a_1^{-1}b_1 \bmod n_1$; alors, l'équation initiale modulo n a d solutions distinctes qui sont

$$x=k,k+n_1,\ldots,k+(d-1)n_1 \bmod n=\{k+qn_1 \bmod n : 0\leq q\leq d-1\}.$$

Chapitre 3

Chiffrements anciens

3.1 Les débuts

Le besoin de communiquer des informations à certaines personnes sans que d'autres puissent prendre connaissance de leur contenu a probablement existé de tout temps. Les affaires militaires et politiques des états, mais aussi le développement du commerce, et bien sûr les raisons privées, ont conduit chaque civilisation à développer des solutions au problème de la *confidentialité*.

La première idée qui vient à l'esprit, pour faire parvenir discrètement un message à quelqu'un, est tout simplement de dissimuler ce message, afin que son existence même soit ignorée des personnes qui ne doivent pas en prendre connaissance. Les techniques utilisées pour dissimuler un message constituent le domaine de la *stéganographie*. Les grecs rasaient le crâne de leurs soldats pour y inscrire un message, bien dissimulé une fois que les cheveux avaient repoussés ; une autre méthode classique, rapportée par Pline l'Ancien, utilise l'encre sympathique, transparente à température normale mais révélée par la chaleur d'une bougie. La stéganographie est encore utilisée de nos jours pour dissimuler une petite quantité d'information dans un fichier image par exemple.

On considère que la stéganographie ne fait pas partie de la cryptographie qui, elle, cherche non pas à dissimuler un message, mais plutôt à le transformer en un message incompréhensible pour une personne non avertie. Le *message clair* devient un *message chiffré* qui en principe garde son secret même s'il tombe dans les mains d'une personne autre que son destinataire. L'action de transformer un message clair en un message chiffré s'appelle *le chiffrement*. Son destinataire, pour obtenir le clair à partir du chiffré, effectue l'opération inverse, *le déchiffrement*. Lorsque une personne tente d'obtenir le message clair à partir du chiffré par ses propres moyens, sans connaitre toutes les données du chiffrement et du déchiffrement, on dit qu'elle *cryptanalyse* ou *décrypte* le message chiffré.

Pendant longtemps, les messages étaient de simples textes écrits dans la langue alphabétique des protagonistes. Nous noterons cet alphabet A, et considèrerons qu'un message est un élément de A^n. Deux méthodes de chiffrement ont dominé les débuts de la cryptographie :

Le chiffrement par transposition agit sur le message en déplaçant ses lettres, i.e. par une permutation sur les positions des lettres du message.

Le chiffrement par substitution agit sur chaque lettre du message par une permutation de l'alphabet A.

La scytale grecque (500 av JC) est un exemple de chiffrement pas transposition. Il s'agit

d'un cylindre autour duquel on entourait une bandelette avant d'écrire le message le long du cylindre. Sur la bandelette déroulée, le texte du message se trouvait écrit en désordre, et il suffisait pour le déchiffrer de posséder une scytale identique autour de laquelle la bandelette était re-enroulée. Une autre méthode de transposition classique consiste à écrire le message clair suivant les lignes d'un carré, puis à le recopier en suivant cette fois les colonnes du carré.

Le chiffre de César, utilisé pour communiquer avec ses armées, opérait un décalage de trois places dans l'alphabet. C'est bien sûr un chiffrement de substitution. Le nombre de permutations d'un alphabet A est de $|A|!$; pour notre alphabet de 26 lettres cela représente plus de 4.10^{26} possibilités ! Toutefois, si on se restreint aux décalages, cela ne fait que 26 possibilités, qui peuvent être essayées successivement dans le cadre d'une cryptanalyse. Pour pallier à cet inconvénient, et augmenter le nombre de permutations à envisager, on procédait souvent de la façon suivante : un "mot-clé" était choisi, par exemple "opération Julius César", puis débarassé de ses répétitions et blancs : "operatinjulsc". Puis on complétait la permutation par les lettres manquantes dans l'ordre de l'alphabet, par exemple ici :

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
o	p	e	r	a	t	i	n	j	u	l	s	c	d	f	g	h	k	m	q	v	w	x	y	z	b

Le chiffrement par substitution a dominé la cryptographie jusqu'à la fin du premier millénaire, et même en Europe jusqu'au 16ème siècle.

3.2 Un peu de formalisme mathématique

On a vu sur les quelques exemples précédents que l'on pouvait distinguer dans une méthode de chiffrement, un algorithme, qui est un principe général d'action, et une clé, qui spécifie ses paramètres. Par exemple, dans le chiffre de César, l'algorithme est le décalage, et la clé est la valeur du pas de décalage, qui vaut 3. Le chiffrement est donc une application :

$$\begin{aligned} E_k : & A^n \to A^n \\ & m \mapsto E_k(m) = c \end{aligned}$$

où k est la clé de chiffrement et appartient à un ensemble K. De même, le déchiffrement est paramétré par une clé k' et on a :

$$\begin{aligned} D_{k'} : & A^n \to A^n \\ & c \mapsto D_{k'}(c). \end{aligned}$$

Pour simplifier, on suppose que le même alphabet est utilisé pour les messages clairs et les messages chiffrés, et que les clés de chiffrement et de déchiffrement parcourent un même ensemble K. Alors, E_k doit être une bijection d'inverse $D_{k'}$, autrement dit on doit avoir :

$$D_{k'}(E_k(m)) = m.$$

La distinction entre l'algorithme et sa clé parait naturelle, pourtant elle a mis longtemps, au cours de l'histoire des chiffrements, à être mise en évidence. Lorsque la cryptographie a été utilisée à plus grande échelle, il est apparu souhaitable, si une méthode de chiffrement devait être utilisée souvent, que la sécurité du chiffrement repose sur le choix et le secret de la clé utilisée plutôt que sur le secret de l'algorithme. De la sorte, si un chiffrement était compromis,

il suffisait de changer la clé sans nécessairement changer de méthode de chiffrement. D'autre part, les partenaires devant se mettre d'accord au préalable sur les données de chiffrement et de déchiffrement, les clés de chiffrement et de déchiffrement représentaient les éléments de cet accord qui devaient être protégés de la curiosité d'autrui.

Ce principe est connu sous le nom de *principe de Kerckhoffs*, d'après le nom d'un officier de l'armée française auteur en 1883 de l'ouvrage *La cryptographie militaire*. Dans ce livre, Kerkhoffs énonce des principes généraux qu'un bon système cryptographique devait suivre et qui sont toujours d'actualité : la sécurité doit reposer sur la clé, le chiffrement et le déchiffrement doivent être rapides à exécuter, la taille de la clé doit être significativement plus courte que celle du message clair.

On peut identifier un alphabet A à m lettres à l'ensemble $\mathbb{Z}/m\mathbb{Z}$, et exploiter les opérations d'addition et de multiplication modulo m pour exprimer certaines substitutions sur A. On utilisera pour notre alphabet à 26 lettres l'identification $a = 0$, $b = 1, \ldots, z = 25$. Ainsi, le décalage de clé k correspond à l'opération $E_k(x) = x + k \bmod 26$, et la bijection réciproque est $D_k(x) = x - k \bmod 26$. Dans ce cas, l'espace des clés est $K = \mathbb{Z}/26\mathbb{Z}$.

Le *chiffrement affine* est une généralisation du chiffrement par décalage. Il transforme $x \in \mathbb{Z}/n\mathbb{Z}$ en $\sigma(x) = ax + b$, où $k = (a, b)$ est la clé du chiffrement. Pour que σ soit une bijection, il faut (et il suffit) que a soit inversible modulo n. En effet, l'application réciproque de $\sigma(x) = ax + b$ est $\sigma^{-1}(y) = a^{-1}(y - b)$. La fonction de déchiffrement est donc aussi affine de clé $k' = (a^{-1}, -ba^{-1})$. L'espace des clés est $K = (\mathbb{Z}/n\mathbb{Z})^* \times \mathbb{Z}/n\mathbb{Z}$. Pour $n = 26$, cela fait $12 * 26 = 312$ clés possibles.

3.3 Le problème de l'échange des clés

Dans les exemples que nous avons évoqués précédemment, les clés de déchiffrement et de chiffrement sont identiques, ou du moins se déduisent l'une de l'autre presque immédiatement. On dit dans ce cas que le système cryptographique est *symétrique* ou encore *à clé privée*. Les protagonistes Alice et Bob doivent donc être tous les deux en possession de cette clé commune. D'autre part, cette clé ne doit en aucun cas tomber aux mains de l'opposant Oscar. Se pose donc le problème initial de *l'échange des clés* : comment Alice et Bob peuvent-ils partager un secret commun (la clé) sans qu'une tierce personne n'en prenne connaissance ? Nous verrons plus tard comment une solution à ce problème est apporté par le concept beaucoup plus récent de *système cryptographique asymétrique*, également appelé *système cryptographique à clé publique*.

3.4 L'espace des clés

Puisque l'algorithme de déchiffrement est paramétré par une clé $k' \in K$ et que, suivant le principe de Kerkhoffs, nous supposons que cet algorithme est connu, il est en principe possible pour Oscar qui tente de cryptanalyser un message chiffré, d'essayer successivement toutes les clés. On parle alors d'*attaque exhaustive*. Pour parer à cette cryptanalyse brutale, il faut donc que l'espace des clés K soit suffisamment grand. Bien sûr, tout dépend des moyens dont Oscar dispose pour passer en revue les valeurs de clés : fait-il ses calculs à la main ou dispose-t-il d'un ordinateur ?

De ce point de vue, le chiffrement par décalage, ou même le chiffrement affine, avec respectivement 26 et 312 clés, paraissent bien faibles. Par contre, un chiffrement par substitution

général, qui spécifie une permutation parmi 26! $\approx 2^{88}$, résisterait à une attaque exhaustive utilisant les moyens informatiques actuels. Nous allons voir que, malgré cela, ce chiffrement est extrêmement faible..

3.5 La cryptanalyse du chiffrement par substitution

Les Arabes ont découvert les premiers la méthode de cryptanalyse du chiffrement de substitution par *l'analyse des fréquences*. Cette méthode est décrite dans le *Manuscrit sur le déchiffrement des messages cryptographiques* écrit par le savant Al-Kindi au IXème siècle :

> Une façon d'élucider un message chiffré, si nous savons dans quelle langue il est écrit, est de nous procurer un autre texte en clair dans la même langue, de la longueur d'un feuillet environ, et de compter alors les apparitions de chaque lettre. Nous appellerons la lettre apparaissant le plus souvent "la première", la suivante "la deuxième", la suivante "la troisième", et ainsi de suite pour chaque lettre figurant dans le texte.
>
> Ensuite, nous nous reporterons au texte chiffré que nous voulons éclaircir et nous relèverons de même ses symboles. Nous remplaçons le symbole le plus fréquent par la lettre "première" du texte clair, le suivant par la "deuxième", et ainsi de suite jusqu'à ce que nous soyons venus à bout de tous les symboles du crytogramme à résoudre.
>
> Abū Yūsuf Ya'qūb ibn Isḥāq al-Kindī

Par exemple, la fréquence d'apparition des lettres dans la langue française est indiquée dans le tableau suivant :

Lettre	%	Lettre	%
A	8.40	N	7.13
B	1.06	O	5.26
C	3.03	P	3.01
D	4.18	Q	0.99
E	17.26	R	6.55
F	1.12	S	8.08
G	1.27	T	7.07
H	0.92	U	5.74
I	7.34	V	1.32
J	0.31	W	0.04
K	0.05	X	0.45
L	6.01	Y	0.30
M	2.96	Z	0.12

Les moyennes dans le texte clair que l'on cherche à décrypter peuvent s'éloigner significativement de celles du tableau, et faire échouer la méthode. Toutefois, on peut s'attendre à une cryptanalyse réussie dès que le texte est assez long. On peut également exploiter, par exemple dans le cas de fréquences de lettres trop proches, d'autres propriétés de même nature, comme la fréquence des paires de lettres contigues.

Cette cryptanalyse repose donc sur la corrélation existant entre les *propriétés statistiques* du texte clair et du texte chiffré. Elle préfigure les méthodes de cryptanalyse statistique modernes.

En Europe, il fallut attendre le XVIème siècle pour que la faiblesse du chiffrement par substitution soit reconnue. Quelques méthodes, peu satisfaisantes, furent développées pour le renforcer : introduire des fautes d'orthographe ou une écriture phonétique, combiner avec un "codage par dictionnaire" d'une partie du vocabulaire, etc. L'objectif bien sûr était de masquer les caractéristiques statistiques du clair (ces chiffrements hybrides n'ont pas empêché Marie Stuart, reine d'Écosse, d'être décapitée en 1587, convaincue de haute trahison contre la couronne d'Angleterre par la cryptanalyse de sa correspondance). Enfin, un chiffrement qui semblait échapper à l'analyse des fréquences fut introduit :

3.6 Le chiffrement de Vigénère

Blaise de Vigénère est un diplomate français, né en 1523. Dans son *Traité des chiffres*, publié en 1586, il propose un chiffrement qui opère par décalage sur l'alphabet, *mais avec une clé variant d'une lettre à la suivante*, au cours d'un cycle répété périodiquement. En termes mathématiques : la clé est $k = (k_1, k_2, \dots, k_s) \in (\mathbb{Z}/26\mathbb{Z})^s$, et, si $m = (a_1, a_2, \dots, a_n, \dots)$ est le clair, son chiffré est

$$c = (a_1 + k_1, \dots, a_s + k_s, a_{s+1} + k_1, \dots, a_{2s} + k_s, a_{2s+1} + k_1, \dots).$$

Le déchiffrement s'effectue suivant le même principe, puisque le clair m est obtenu à partir du chiffré $c = (b_1, b_2, \dots)$ par soustraction de la clé :

$$m = (b_1 - k_1, \dots, b_s - k_s, b_{s+1} - k_1, \dots, b_{2s} - k_s, b_{2s+1} - k_1, \dots).$$

Sur un tel chiffrement, on ne pouvait plus faire d'analyse des fréquences suivant la méthode préconisée par Al-Kindi. En effet, le changement de clé perturbe complètement la fréquence des lettres dans le chiffré.

Toutefois, *si on connait la longueur s de la clé*, et si le texte est assez long, il y a bien un moyen : c'est d'analyser les fréquences parmi les lettres dont les positions sont identiques modulo s. En effet, celles-ci sont soumises à la même clé de décalage, donc sont distribuées suivant le profil de fréquences du clair.

Charles Babbage, à Londres, par ailleurs inventeur de la première machine à calculer, réussit à cryptanalyser le chiffrement de Vigénère probablement en 1854 (mais sans publier sa découverte). Friedrich Wilhem Kasiski, un officier prussien retraité, parvint au même résultat en 1863 dans son ouvrage *Écriture secrète et art du déchiffrement*. Leur méthode est la suivante : la répétition d'une même suite de lettres, par exemple un trigramme donné, dans le chiffré, indique très probablement un trigramme fréquent du clair, qui se trouve en position d'être chiffré avec le même trio de clé. Si c'est le cas, les différences de positions entre ces trigrammes sont des multiples de s. Ainsi, avec un peu de chance, on obtient s en calculant le pgcd de ces différences.

3.7 Le masque jetable et la sécurité inconditionelle

La cryptanalyse du chiffrement de Vigénère n'est possible que si la clé k utilisée est relativement courte par rapport à la longueur du message clair. Les cryptographes ont donc proposé d'utiliser ce mode de chiffrement avec une clé aussi longue que le message. Bien sûr, cela pose des problèmes techniques importants dans la pratique, puique les protagonistes doivent au

préalable s'accorder sur la valeur de la clé. Il est tentant alors de choisir une clé facile à retenir ou à retrouver, comme une phrase, ou un extrait d'un livre connu. Il est vite apparu que, si la clé choisie a des caractéristiques statistiques trop prévisibles, elle compromet la sécurité du chiffrement.

D'autre part, l'avènement du télégraphe a transformé la façon dont on transcrivait un message. Une information transmise par cette nouvelle technologie était au préalable codée avec seulement deux signes (utilisant le code Baudot, ancêtre du code ASCII, qui codait un caratère sur 5 "bits"). Autrement dit, un message est envisagé comme un mot binaire, un élément de $\{0,1\}^n$, ou encore de $(\mathbb{Z}/2\mathbb{Z})^n$.

Gilbert Vernam était ingénieur au AT&T Bell Labs quand il conçut en 1917 un système automatique de chiffrement par ajout d'une clé binaire au moyen de cartes perforées et d'un téléscripteur électrique. Il déposa un brevet pour son invention en 1919. En termes mathématiques, le *chiffrement de Vernam* prend la forme suivante :

$$k \in (\mathbb{Z}/2\mathbb{Z})^n, \quad E_k : (\mathbb{Z}/2\mathbb{Z})^n \to (\mathbb{Z}/2\mathbb{Z})^n \quad \text{et } D_k = E_k.$$
$$m \mapsto m \oplus k$$

L'addition $\oplus$ dans $(\mathbb{Z}/2\mathbb{Z})^n$ est exécutée modulo 2, coordonnée par coordonnée : par exemple,

$$111100 \oplus 110101 = 001001$$

En informatique on l'appelle "xor". Cette addition est analogue à l'addition de vecteurs de $\mathbb{R}^n$.

Un peu plus tard, Joseph Mauborgne, un capitaine de l'armée américaine, proposa de renforcer le chiffrement de Vernam par la condition que la clé k soit choisie *aléatoirement* dans $(\mathbb{Z}/2\mathbb{Z})^n$ (dans le brevet elle était périodique !). Dans l'article *Cipher printing telegraph systems for secret wire and radio telegraphic communications* de 1926, Vernam qualifie son chiffrement d"'absolument incassable" si *la clé est choisie aléatoirement et n'est jamais réutilisée (on parle alors de masque jetable).*

En effet, supposons qu'un même clé k soit utilisée pour chiffrer deux messages clairs x_1 et x_2. On a donc

$$c_1 = x_1 \oplus k \quad \text{et} \quad c_2 = x_2 \oplus k.$$

Si Oscar intercepte les messages chiffrés c_1 et c_2, il peut calculer

$$c_1 \oplus c_2 = x_1 \oplus k \oplus x_2 \oplus k = x_1 \oplus x_2.$$

Ainsi, en additionnant les deux chiffrés, il annule l'effet des masques et obtient $x_1 \oplus x_2$. Bien sûr, à partir de $x_1 \oplus x_2$ il ne peut pas obtenir directement x_1 et x_2. Mais il peut exploiter les propriétés statistiques des clairs pour obtenir de l'information sur ceux-ci. Par exemple, s'il connait une partie de x_1 il peut en déduire la partie de x_2 correspondante.

Claude Shannon (AT&T Bell Labs) est le fondateur de la théorie de l'information. Dans un article publié en 1949, *Communication Theory of Secrecy Systems*, il montre que le masque jetable guarantit une *sécurité inconditionnelle*, au sens de sa toute nouvelle théorie basée sur les probabilités, et que celle-ci ne peut être guarantie par un système dont les clés sont significativement plus petites.

Il va de soi que le masque jetable, est assez peu praticable, en particulier dans le cas de chiffrements multiples. En particulier, on ne sait pas comment produire du "vrai aléa". Il a toutefois été utilisé, en particulier pour sécuriser le téléphone rouge entre Washington et Moscou durant la guerre froide.

3.8 Les machines à chiffrer

Après la première guerre mondiale, des machines à chiffrer électromécaniques ont vu le jour. En particulier, la machine Enigma est mise au point par deux inventeurs allemands, Arthur Scherbius et Richard Ritter, brevetée en 1918, et vendue à l'armée allemande en 1925. Le chiffrement par Enigma effectuait une permutation de l'alphabet, mais cette permutation était modifié à chaque lettre, suivant un système complexe de connexions et de rotors. Autrement dit, le chiffrement avait la forme suivante :

$$E_k : A^n \to A^n \quad \text{et } D_k = E_k.$$
$$m = (a_1, \ldots, a_n) \mapsto c = (\sigma_1(a_1), \ldots, \sigma_n(a_n)).$$

Plus précisément, la clé k indiquait une position initiale du système, définissant une permutation initiale de l'alphabet. La frappe d'une lettre provoquait la rotation d'un rotor, laquelle induisait un changement de permutation. Le nombre de rotors (qui a varié de 3 à 8) et l'ordre d'installation des rotors rendait le nombre de permutations possibles véritablement gigantesque, et la position initiale n'était retrouvée qu'au bout d'un très grand nombre d'utilisations. De plus, la structure était telle que la même machine pouvait être utilisée pour le chiffrement et pour le déchiffrement.

Les polonais ont initié la cryptanalyse d'Enigma en 1930. À ce moment, d'une part l'architecture de la machine est dans sa forme la plus simple, et d'autre part les opérateurs ont l'habitude de chiffrer deux fois successivement la clé de session. Le polonais Rejewski comprit les avantages qu'il pouvait tirer de cela : en effet, le cryptanalyste est alors en mesure de limiter une recherche exhaustive aux clés compatibles. Pour être à même de tester toutes les possibilités restantes, les polonais construisent en 1938 la première *bombe*, véritable ordinateur électromécanique, et grâce à elle parviennent à décrypter la plupart des messages chiffrés allemands. Puis la Pologne est envahie.. mais les polonais réussisent à faire parvenir leurs avancées aux alliés. La cryptanalyse d'Enigma devient l'affaire des anglais et des américains, dans le manoir de Bletchley Park. L'une des équipes de cryptologues est dirigée par Alan Turing.

Ultérieurement, Enigma fut significativement améliorée par les allemands, et d'autre part la clé de session ne fut plus chiffrée deux fois. Malgré cela, la recherche de mots probables dans le clair, quelques erreurs de manipulation des allemands, et bien d'autres techniques cryptographiques, jointes à la puissances des bombes cryptographiques conçues par Alan Turing et son équipe, venaient à bout du déchiffrement des messages allemands.

Chapitre 4

La cryptographie symétrique moderne

Dans ce chapitre, nous abordons la période postérieure à 1950. Sauf mention du contraire, un message clair est indentifié à un élément de $(\mathbb{Z}/2\mathbb{Z})^n$.

4.1 Introduction

L'une des difficultés majeures du chiffrement par masque jetable, c'est de générer une clé véritablement aléatoire. Dans les applications pratiques, on la remplace par une *suite binaire pseudo-aléatoire* calculée (de façon déterministe) à partir d'un petit nombre de bits véritablement aléatoires. On aimerait idéalement que cette suite pseudo-aléatoire soit *indistinguable* d'une suite aléatoire. Un tel système prend le nom de *générateur de nombre pseudo-aléatoires* et a de nombreuses applications pratiques. On va en étudier quelques exemples.

4.2 Les générateurs de nombres pseudo-aléatoires

Un générateur de nombres pseudo-aléatoires est donc une application :

$$\begin{aligned} S : (\mathbb{Z}/2\mathbb{Z})^t &\to (\mathbb{Z}/2\mathbb{Z})^{\mathbb{N}} \\ k &\mapsto S(k) = s = (s_0, s_1, s_2, s_3, \dots) \end{aligned}$$

On dit que k est son *initialisation*, ou sa *graine*, ou (pour utilisation en crypto) sa *clé*. On demande que les bits successifs de $S(k)$ se calculent algorithmiquement et rapidement à partir de k, et que $S(k)$ "ressemble" autant que possible à une suite aléatoire.

Plus précisément, on souhaite que $S(k)$ soit *calculatoirement indistinguable d'une suite aléatoire*.

Introduisons d'abord un peu de vocabulaire :

Définition 4.2.1. Soit $s = (s_0, s_1, s_2, s_3, \dots) \in (\mathbb{Z}/2\mathbb{Z})^{\mathbb{N}}$.

1. On dit que s est *périodique* de période T si $s_i = s_{i+T}$ pour tout $i \geq 0$.
2. On dit que s est *ultimement périodique* de période T si $s_i = s_{i+T}$ pour tout $i \geq i_0$. Alors $(s_0, s_1, \dots, s_{i_0-1})$ s'appelle la *prépériode* de s.
3. On dit que s est *équilibrée* si la proportion de 0 (et donc de 1) dans s est de $1/2$.

4. Un *mot* extrait de s (de longueur u) est un mot binaire de la forme $(s_i, s_{i+1}, \dots, s_{i+u-1})$.

Remarque 4.2.2. On remarque que, si s est périodique de période T, alors tout multiple de T est aussi une période. En effet, si s est périodique de période T, alors $s_{i+2T} = s_{(i+T)+T} = s_{i+T} = s_i$, donc s est aussi périodique de période $2T$; en itérant ce raisonnement, on voit que s est périodique de période kT pour tout entier $k \geq 1$.

Il n'y a donc pas unicité de la période d'une suite ; par contre, il y a toujours une unique *plus petite période* qui divise toutes les autres. Quand on parlera de LA période d'une suite, on sous-entendra cette plus petite période.

Revenons maintenant aux générateurs pseudo-aléatoires. Intuitivement, pour que le générateur S soit satisfaisant, il est nécessaire qu'il satisfasse à certains critères : s'il engendre une suite s ultimement périodique, il faut que sa plus petite période soit grande (exponentielle en t la taille de la graine) ; s devrait être équilibrée, et la fréquence des mots de petite longueur devrait être proche de l'uniforme (même proportion de 00, 01, 10, 11 par exemple). Il existe un grand nombre d'autres *critères statistiques* auxquels on peut soumettre un générateur de nombres pseudo-aléatoires, que nous ne pouvons pas aborder dans le cadre de ce cours.

Remarque 4.2.3. On peut bien sûr introduire les générateurs de nombres pseudo-aléatoires sur des anneaux plus généraux que $\mathbb{Z}/2\mathbb{Z}$; par exemple $\mathbb{Z}$ ou $\mathbb{Z}/m\mathbb{Z}$.

4.3 Les registres à décalage linéaires

Ce sont les plus simples des générateurs pseudo-aléatoires que l'on peut imaginer. Même s'ils ne sont pas satisfaisants en tant que tels pour les applications cryptographiques (pour des raisons que nous ne pouvons pas développer ici), ils servent de briques de base pour beaucoup de générateurs utilisés en cryptographie, d'où leur importance.

Définition 4.3.1. Un *registre à décalage linéaire (LFSR)*, est une application :

$$\begin{aligned} S : (\mathbb{Z}/2\mathbb{Z})^t &\to (\mathbb{Z}/2\mathbb{Z})^{\mathbb{N}} \\ k &\mapsto S(k) = s = (s_0, s_1, s_2, s_3, \dots) \end{aligned}$$

définie par des coefficients $(a_0, a_1, \dots, a_{t-1}) \in (\mathbb{Z}/2\mathbb{Z})^t$ tels que :

$$\begin{cases} (s_0, \dots, s_{t-1}) = (k_0, \dots, k_{t-1}) \\ s_{i+t} = a_0 s_i + a_1 s_{i+1} + \dots a_{t-1} s_{i+t-1} \text{ pour tout } i \geq 0. \end{cases}$$

Le nombre t s'appelle la *longueur* du registre.

Exemple 4.3.2. Prenons $t = 3$, et $(a_0, a_1, a_2) = (1, 1, 0)$. Si l'initialisation est $k = (1, 0, 0)$ on obtient pour suite s :

$$1001011100101110010110 0 1 0 \dots$$

qui est périodique de période 7. On remarque que l'initialisation 100 se répète en position 7.

Proposition 4.3.3. On a les propriétés suivantes :

1. Un registre à décalage linéaire de longueur t engendre 2^t suites distinctes.
2. Toute suite engendrée par un LFSR de longueur t est ultimement périodique de période $T \leq 2^t - 1$.

3. Si $a_0 \neq 0$, toute suite engendrée est périodique.

Démonstration. 1. Un LFSR engendre 2^t suites distinctes qui sont en correspondance avec les 2^t initialisations possibles.

2. On remarque que l'initialisation $(s_0, \ldots, s_{t-1}) = (0, 0, \ldots, 0)$ engendre la suite nulle $s = 000000\ldots$. On remarque également que l'on peut supposer $a_0 \neq 0$, sinon, quitte à ignorer les premiers coefficients de s, un LFSR de longueur plus petite et vérifiant $a_0 \neq 0$ engendre s.

Si $a_0 \neq 0$, il existe une récurrence linéaire qui "descent" la suite s. En effet, on a

$$s_{i+t} = a_0 s_i + a_1 s_{i+1} + \ldots a_{t-1} s_{i+t-1}$$

qui entraine

$$a_0 s_i = -a_1 s_{i+1} - \cdots - a_{t-1} s_{i+t-1} + s_{i+t}$$

soit (en binaire)

$$s_i = a_1 s_{i+1} + \cdots + a_{t-1} s_{i+t-1} + s_{i+t}.$$

Cela prouve que, si un mot de s de longueur t vaut 0^t, la suite est nulle dès son initialisation.

Supposons maintenant que $s \neq 0$. Alors les mots successifs $m_i = (s_i, s_{i+1}, \ldots s_{i+t-1})$ appartiennent à $(\mathbb{Z}/2\mathbb{Z})^t$ et sont différents de 0^t. Il y en a donc au plus $2^t - 1$ deux à deux distincts. Donc, parmi $\{m_0, m_1, \ldots, m_{2^t-1}\}$ il y a au moins deux mots égaux. Supposons $0 \leq i < j \leq 2^t - 1$ tels que $m_i = m_j$. Comme la suite s à partir d'un rang donné k ne dépend que du mot m_k, elle ne peut être que périodique de période $(j - i)$ à partir du rang i. Or $(j - i) \leq 2^t - 1$, donc la suite est ultimement périodique de période $T \leq 2^t - 1$.

3. Si $a_0 \neq 0$, par le procédé de descente expliqué plus haut, la période se propage jusqu'au début de s □

Définition 4.3.4. Une suite s est dite *maximale* si s est engendrée par un LFSR de longueur t et si s est périodique de période $2^t - 1$, et n'admet pas de période plus petite.

Proposition 4.3.5. Si s est une suite maximale de période $2^t - 1$, alors un mot de longueur $u \leq t$ et différent de 0^u apparait exactement 2^{t-u} fois dans une période ; le mot 0^u apparait, lui, $2^{t-u} - 1$ fois. En particulier, le nombre de 1 dans une période de s est 2^{t-1} et le nombre de 0 est $2^{t-1} - 1$.

Démonstration. D'après ce qui précède, si s est de période maximale, les mots $m_0, \ldots, m_{2^t-2}$ de longueur t sont deux à deux distincts et non nuls, donc apparaissent exactement une fois par période. Si $x \neq 0^u$ est de longueur $u \leq t$, il est le début de exactement 2^{t-u} mots de longueur t. □

Proposition 4.3.6. Si S est un LFSR tel qu'il existe une initialisation k telle que $s = S(k)$ soit maximale, alors c'est vrai pour toutes les suites non nulles engendrées par ce LFSR, et de plus celles-ci sont obtenues à partir de s par troncation.

Démonstration. Si s est maximale, cela signifie qu'elle est de période $2^t - 1$, donc que ses mots $\{m_0, \ldots, m_{2^t-2}\}$ sont distincts et non nuls, donc parcourent tous les éléments de $\{0, 1\}^t \backslash \{0\}$. Si on supprime ses i premiers termes $s_0, s_1, \ldots, s_{i-1}$, on obtient une nouvelle suite qui commence par le mot m_i, donc c'est $S(m_i)$. □

4.4 Les registres à décalage non linéaires

On peut définir des registres à décalage non linéaires, i.e. associés à une fonction de transition

$$s_{i+t} = f(s_i, s_{i+1}, \dots, s_{i+t-1})$$

non linéaire. Pour les mêmes raisons que précédemment ils sont périodiques de période inférieure ou égale à 2^t.

Exemple 4.4.1. On peut prendre $f(s_i, s_{i+1}, s_{i+2}) = s_i s_{i+1} + s_{i+2}$.

4.5 Les générateurs linéaires congruentiels

Ils sont définis sur $\mathbb{Z}/m\mathbb{Z}$. Si on veut, on peut facilement les transformer en générateurs binaires (exemple : garder seulement le bit de poids faible).

Définition 4.5.1. Un *générateur linéaire congruentiel*, est une application :

$$\begin{aligned} S : \mathbb{Z}/m\mathbb{Z} &\to (\mathbb{Z}/m\mathbb{Z})^{\mathbb{N}} \\ k &\mapsto S(k) = s = (s_0, s_1, s_2, s_3, \dots) \end{aligned}$$

définie par des coefficients $(a, b) \in (\mathbb{Z}/m\mathbb{Z})^2$ tels que :

$$s_0 = k \text{ et } s_{i+1} = as_i + b \text{ pour tout } i \geq 0.$$

Une suite engendrée par un générateur linéaire congruentiel est ultimement périodique, de période au plus m. Si $a \in (\mathbb{Z}/m\mathbb{Z})^*$, la suite est périodique (i.e. il n'y a pas de pré-période).

4.6 Utilisation en cryptographie : le chiffrement à flot

On parle de chiffrement à flot si on chiffre un message clair x binaire par une suite $s = S(k)$ engendrée par un générateur binaire pseudo aléatoire de clé k, suivant

$$E_k(x) = x \oplus S(k).$$

De cette façon, on imite le chiffrement par masque jetable. Le générateur S mis en oeuvre étant supposé connu de tous, la clé du chiffrement coincide avec son initialisation et doit être échangée (par Alice et Bob dans le schéma traditionnel) au préalable. Le déchiffrement D_k est identique au chiffrement.

Une même clé k ne doit pas être utilisée deux fois pour chiffrer (voir le paragraphe du chapitre précédent sur le maque jetable), et, à cause de cela, on ne doit pas dépasser la période de $S(k)$. Notons que, si la période est de l'ordre de 2^t, où t est la longueur de la clé, on peut quand même chiffrer un message binaire de taille conséquente : par exemple un méga octet (1 MO) correspond à environ 2^{23} chiffres binaires soit une clé de taille binaire 23.

Les LFSR ont été utilisés pour le chiffrement jusqu'en 1969, lorsque Berlekamp a découvert qu'ils n'étaient pas cryptographiquement sûrs, même si la longueur et les coefficients sont secrets. Aujourd'hui, on les utilise comme briques élémentaires de systèmes plus complexes, mettant en oeuvre des fonctions non linéaires. Ce type de chiffrement est par exemple utilisé à l'heure actuelle en téléphonie mobile (algorithmes de la famille A5, RC4).

Il existe des générateurs de nombres pseudo-aléatoires cryptographiquement sûrs basés sur de l'arithmétique plus complexe, exemple : Blum Blum Shub mais ils sont beaucoup plus lents.

4.7 Chiffrement par blocs

Un algorithme symétrique de chiffrement par blocs est défini par une fonction de chiffrement :

$$\begin{aligned} E_k : (\mathbb{Z}/2\mathbb{Z})^n &\to (\mathbb{Z}/2\mathbb{Z})^n \\ x &\mapsto E_k(x) \end{aligned}$$

opérant sur des blocs de taille n fixée. Dans la pratique, n est de l'ordre de 100. Celle-ci ne doit pas être trop petite pour éviter une attaque *par dictionnaire*, qui consiste à stocker des couples $(x, E_k(x))$. La taille des clés doit être du même ordre de grandeur pour éviter une attaque exhaustive.

Envisageons quelques possibilités pour la fonction E_k :

– L'ajout de clé

$$\begin{aligned} (\mathbb{Z}/2\mathbb{Z})^n &\to (\mathbb{Z}/2\mathbb{Z})^n \\ x &\mapsto x \oplus k \end{aligned}$$

comme dans le chiffrement de Vernam. Le problème majeur est que la clé k ne peut pas être utilisée plus d'une fois.

– Une transformation linéaire de matrice inversible $A \in \mathrm{GL}(n, \mathbb{Z}/2\mathbb{Z})$

$$\begin{aligned} E_A : (\mathbb{Z}/2\mathbb{Z})^n &\to (\mathbb{Z}/2\mathbb{Z})^n \\ x &\mapsto xA \end{aligned}$$

où A est la clé du chiffrement. La fonction de déchiffrement est la multiplication par A^{-1} l'inverse de A.

On va voir que, si l'attaquant Oscar, qui ne connait pas la clé A, peut obtenir suffisamment de couples (messages clairs x_i, messages chiffrés $y_i = E_A(x_i)$), alors il peut calculer A. En effet, puisque $x_i A = y_i$, on a l'identité matricielle :

$$XA = Y$$

où X (respectivement Y) est la matrice dont les lignes sont constituées des x_i (respectivement des y_i). Avec n messages tels que la matrice X soit inversible, il peut en déduire la clé

$$A = X^{-1}Y.$$

Donc cet algorithme de chiffrement n'est pas satisfaisant par lui-même. Remarquons que le chiffrement par transposition traditionnel est un cas particulier de celui-ci (dans lequel la matrice A est une *matrice de permutation*).

– Une application bijective E non linéaire, voire aléatoire, spécifiée par la donnée de tous les $E(x)$. On ne peut pas envisager d'utiliser une telle fonction sur des blocs entiers de taille $n \simeq 100$ car il faudrait mémoriser $n2^n$ chiffres binaires. Par contre on peut appliquer ce type de tranformation à des sous-blocs de taille $b \simeq 10$.

Ce type de chiffrement, associé à une petite valeur de b, est analogue au chiffrement par substitution traditionnel. On a vu qu'il risque de révéler des informations sur la structure statistique du clair, qui peuvent être exploités dans une cryptanalyse.

On voit que, pris séparément, chacune de ces fonctions de chiffrement a des inconvénients. Pour définir une fonction de chiffrement satisfaisante, il est nécessaire de composer successivement un certain nombre de transformations de l'un de ces trois types, en alternance.

Les chiffrements symétriques modernes sont conçus de cette façon. Ils sont définis par une *fonction de tour* qui exécute au moins une transformation de chacun des ces types, et itèrent plusieurs tours successifs sur le clair (on dit que ce sont des algorithmes itératifs). Pour déchiffrer, il faut appliquer sur le chiffré autant de fois la fonction de tour inverse.

Exemples :

- L'algorithme DES fut le standard du chiffrement symétrique adopté par le NIST de 1977 jusqu'en 2000. Issu d'un algorithme utilisé par IBM (Lucifer) il fut modifé par la NSA en vue de sa standardisation. Il opère sur des blocs de 64 bits avec des clés de 56 bits. Il n'est plus considéré cryptographiquement sûr, essentiellement à cause de la taille de sa clé qui est accessible à une attaque exhaustive informatique.
- En 1997, le NIST lance un concours international pour définir un nouveau standard de chiffrement symétrique l'AES (Advanced Encryption Standard). Parmi plusieurs propositions, l'algorithme RIJNDAEL proposé par deux chercheurs belges est choisi et remplace DES à partir de 2000. Il opère sur des blocs de taille 128, avec une clé au choix de 128,192, ou 256 bits. Sa structure fait intervenir les opérations d'addition et de multiplication dans *le corps fini à 256 éléments.*
- Il y en a bien d'autres. Le logiciel PGP (Pretty Good Privacy) qui propose des outils cryptographiques à usage privé, intègre les algorithmes IDEA, TripleDES, CAST5, Blowfish, AES, Twofish.

Chapitre 5

Compléments d'arithmétique

5.1 Le théorème des restes chinois

Pour introduire notre propos, commençons par une petite histoire :

> *Une bande de 17 pirates possède un trésor constitué de pièces d'or d'égale valeur. Ils projettent de se les partager également, et de donner le reste au cuisinier chinois. Celui-ci recevrait alors 3 pièces. Mais les pirates se querellent, et six d'entre eux sont tués. Un nouveau partage donnerait au cuisinier 4 pièces. Dans un naufrage ultérieur, seuls le trésor, six pirates et le cuisinier sont sauvés, et le partage donnerait alors 5 pièces d'or à ce dernier. Quelle est la fortune minimale que peut espérer le cuisinier s'il décide d'empoisonner le reste des pirates ?*

Traduisons ce problème en équations. Si T est le nombre de pièces qui constituent le trésor,

$$\begin{cases} T = 3 \bmod 17 \\ T = 4 \bmod 11 \\ T = 5 \bmod 6 \end{cases} \tag{5.1}$$

et on cherche à calculer T à partir de ce système de congruences. Nous allons voir que le théorème suivant permet de résoudre ce problème :

Théorème 5.1.1. (Le théorème des restes chinois, version 1) Soit m et n deux entiers premiers entre eux. Soit $1 = mu + nv$ une relation de Bézout entre m et n. Alors, pour tout entiers a, b, x, on a :

$$x = a \bmod m \text{ et } x = b \bmod n \iff x = (anv + bmu) \bmod mn.$$

Montrons d'abord comment utiliser ce théorème pour calculer la valeur du trésor des pirates. On considère les deux premières congruences : $T = 3 \bmod 17$ et $T = 4 \bmod 11$. Les entiers $m := 17$ et $n := 11$ sont bien premiers entre eux, et l'algorithme d'Euclide étendu (que nous ne détaillons pas ici) conduit à la relation de Bézout $2 * 17 - 3 * 11 = 1$. D'après le théorème,

$$\begin{aligned} T = 3 \bmod 17 \text{ et } T = 4 \bmod 11 &\iff T = (3 * (-3 * 11) + 4 * (2 * 17)) \bmod 17 * 11 \\ &\iff T = 37 \bmod 187. \end{aligned}$$

Il nous reste à résoudre le système de congruences : $T = 37 \bmod 187$ et $T = 5 \bmod 6$. Comme 187 et 6 sont premiers entre eux, on peut appliquer à nouveau le théorème 5.1.1. On a pour relation de Bézout $187 - 31 * 6 = 1$ donc

$$\begin{aligned} T = 37 \bmod 187 \text{ et } T = 5 \bmod 6 &\iff T = (37 * (-31 * 6) + 5 * (187)) \bmod 187 * 6 \\ &\iff T = 785 \bmod 1122 \end{aligned}$$

donc le trésor contient au minimum 785 pièces.

Démonstration du Théorème 5.1.1 : Comme $mu+nv = 1$, on a le tableau suivant de conguences :

	$\bmod m$	$\bmod n$
nv	1	0
mu	0	1
$anv + bmu$	a	b

D'autre part, l'implication suivante est vraie :

$$x = y \bmod mn \Longrightarrow \begin{cases} x = y \bmod m \\ x = y \bmod n \end{cases}$$

En conséquence, si $x = (anv + bmu) \bmod mn$, alors $x = (anv + bmu) \bmod m$, donc (par le tableau) $x = a \bmod m$. De même, $x = (anv + bmu) \bmod n = b \bmod n$. On a donc démontré l'implication $2 \Rightarrow 1$. Il reste à démontrer l'implication $1 \Rightarrow 2$.

Supposons donc que $x = a \bmod m$ et $x = b \bmod n$. Posons $c = anv + bmu$. Alors, modulo m, $x - c = a(1 - nv) - bmu = amu - bmu = 0 \bmod m$. De même, $x - c = 0 \bmod n$. Donc m et n divisent $x - c$; donc $\text{ppcm}(m, n)$ divise $x - c$. Mais, comme $\text{pgcd}(m, n) = 1$, cela implique que $\text{ppcm}(m, n) = mn$, donc finalement mn divise $x - c$, soit $x = c \bmod mn$. □

Théorème 5.1.2. (Le théorème des restes chinois, version 2) Soit m et n deux entiers premiers entre eux. L'application

$$\begin{aligned} f : \mathbb{Z}/mn\mathbb{Z} &\to \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z} \\ x \bmod mn &\mapsto (x \bmod m, x \bmod n) \end{aligned}$$

est une bijection, dont l'application réciproque est définie par :

$$\begin{aligned} g : \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z} &\to \mathbb{Z}/mn\mathbb{Z} \\ (a \bmod m, b \bmod n) &\mapsto (anv + bmu) \bmod mn \end{aligned}$$

De plus, les applications f et g sont *compatibles avec les opérations d'addition et de multiplication.* On dit que ce sont des *isomorphismes d'anneaux.*

Démonstration. En effet, le théorème 5.1.1 montre que tout élément $(a, b) \in \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}$ possède un unique antécédent par f. □

Remarque 5.1.3. Le fait que les applications f et g soient compatibles avec les opérations est d'une grande importance pratique. En effet, cela signifie que, pour exécuter un calcul algébrique modulo mn, il suffit de l'exécuter modulo m, puis modulo n, puis d'appliquer g au couple de résultats.

Exemple 5.1.4. Nous voulons calculer 14^4 mod 143. On a $143 = 11 * 13$ et une relation de Bézout entre 11 et 13 est : $6 * 11 - 5 * 13 = 1$. On effectue le calcul modulo 11 et modulo 13 :

$$\begin{cases} 14^4 = 3^4 = 4 \bmod 11 \\ 14^4 = 1^4 = 1 \bmod 13 \end{cases}$$

Soit, avec les notations du théorème 5.1.2, $f(14^4 \bmod 143) = (4 \bmod 11, 1 \bmod 13)$. Il suffit maintenant de calculer $g((4 \bmod 11, 1 \bmod 13))$. On a : $g((4 \bmod 11, 1 \bmod 13)) = 4 * (-5 * 13) + 1 * (6 * 11) = 92 \bmod 143$. Donc, $14^4 = 92 \bmod 143$.

Exemple 5.1.5. Nous voulons calculer l'inverse (s'il existe) de 17 modulo 143. Un calcul rapide montre que 17 est inversible modulo 11 d'inverse 2, et que 17 est inversible modulo 13 d'inverse 10. Alors, comme une relation de Bézout entre 11 et 13 est : $6 * 11 - 5 * 13 = 1$, on obtient l'inverse de 17 mod 143 en calculant $(-5 * 13) * 2 + (6 * 11) * 10 \bmod 143$ qui vaut 101 mod 143.

5.2 La fonction φ d'Euler

Définition 5.2.1. La fonction φ d'Euler est définie, pour tout $n \geq 1$ entier par :

$$\varphi(n) = |\{a : 1 \leq a \leq n \text{ et } \operatorname{pgcd}(a, n) = 1\}|.$$

On a déjà vu au chapitre 2 que :

Théorème 5.2.2.

$$|(\mathbb{Z}/n\mathbb{Z})^*| = \varphi(n).$$

Proposition 5.2.3. La fonction φ d'Euler vérifie les propriétés suivantes :

1. $\varphi(1) = 1$
2. Pour tout p premier, et tout $k \geq 1$, $\varphi(p^k) = p^k - p^{k-1}$.
3. Pour tout m, n tels que $\operatorname{pgcd}(m, n) = 1$, $\varphi(mn) = \varphi(m)\varphi(n)$.

Démonstration. 1. est évident. Montrons 2. Pour cela, considérons $b \in \{1, \ldots, p^k\}$ tel que $\operatorname{pgcd}(b, p^k) > 1$. Puisque b n'est pas premier à p^k, ils ont un diviseur commun différent de 1 ; cela signifie que b est divisible par p. Alors, $b = pq$ avec $q \in \{1, \ldots, p^{k-1}\}$. Donc b peut prendre p^{k-1} valeurs, donc $\varphi(p^k) = p^k - p^{k-1}$.

La propriété 3 est une conséquence du théorème chinois 5.1.2. En effet, d'après celui-ci, si m et n sont premiers entre eux, si $a \in \{1, \ldots, mn\}$, alors a est inversible modulo mn si et seulement si a est inversible modulo m et modulo n. On a donc une correspondance bijective entre $(\mathbb{Z}/mn\mathbb{Z})^*$ et $(\mathbb{Z}/m\mathbb{Z})^* \times (\mathbb{Z}/n\mathbb{Z})^*$. Par conséquent, les cardinaux de ces ensembles sont égaux, ce qui démontre que $\varphi(mn) = \varphi(m)\varphi(n)$. □

Remarque 5.2.4. La proposition 5.2.3 permet de calculer aisément $\varphi(n)$, si on connait la décomposition en produit de puissances de nombres premiers de n. En effet, si $n = p_1^{k_1} \ldots p_s^{k_s}$, en appliquant successivement les propriétés 3. et 2., on obtient :

$$\begin{aligned} \varphi(n) &= \varphi(p_1^{k_1}) \ldots \varphi(p_s^{k_s}) \\ &= (p_1^{k_1} - p_1^{k_1 - 1}) \ldots (p_s^{k_s} - p_s^{k_s - 1}) \\ &= n(1 - \frac{1}{p_1}) \ldots (1 - \frac{1}{p_s}). \end{aligned}$$

Exemple 5.2.5. Calulons $\varphi(19440)$. On a $19440 = 3^5 * 4^2 * 5$ donc $\varphi(19440) = \varphi(3^5)\varphi(4^2)\varphi(5) = (3^5 - 3^4) * (4^2 - 4) * (5 - 1) = 5184$.

Remarque 5.2.6. En particulier, si p est premier, $\varphi(p) = p - 1$. On retrouve le fait que tout élément non nul de $\mathbb{Z}/p\mathbb{Z}$ est inversible.

5.3 Le théorème d'Euler et le petit théorème de Fermat

Théorème 5.3.1. (Théorème d'Euler) Pour tout entiers $n \geq 1$ et a tels que $\text{pgcd}(a, n) = 1$,

$$a^{\varphi(n)} = 1 \bmod n.$$

Démonstration. Parce que $\text{pgcd}(a, n) = 1$, on sait que a est inversible modulo n. Alors, l'application :

$$\begin{aligned}(\mathbb{Z}/n\mathbb{Z})^* &\to (\mathbb{Z}/n\mathbb{Z})^* \\ x &\mapsto ax\end{aligned}$$

est une bijection. En effet, elle possède une application réciproque qui est la multiplication par a^{-1} :

$$\begin{aligned}(\mathbb{Z}/n\mathbb{Z})^* &\to (\mathbb{Z}/n\mathbb{Z})^* \\ x &\mapsto a^{-1}x.\end{aligned}$$

Les ensembles suivants sont donc identiques :

$$(\mathbb{Z}/n\mathbb{Z})^* = \{ax : x \in (\mathbb{Z}/n\mathbb{Z})^*\}.$$

On considère maintenant le produit P des éléments de $(\mathbb{Z}/n\mathbb{Z})^*$. On a donc :

$$P = \prod_{x \in (\mathbb{Z}/n\mathbb{Z})^*} x = \prod_{x \in (\mathbb{Z}/n\mathbb{Z})^*} ax = a^{\varphi(n)}P.$$

où on a utilisé que $|(\mathbb{Z}/n\mathbb{Z})^*| = \varphi(n)$. Comme $P \neq 0$, on en déduit que $a^{\varphi(n)} = 1 \bmod n$. □

Corollaire 5.3.2. Pour tout p premier et a tel que $\text{pgcd}(a, p) = 1$,

$$a^{p-1} = 1 \bmod p.$$

Démonstration. On applique le théorème d'Euler à $n = p$, avec $\varphi(p) = p - 1$. □

Corollaire 5.3.3. (Le petit théorème de Fermat) Pour tout p premier et pour tout entier a,

$$a^p = a \bmod p.$$

Démonstration. On a déjà vu que, si $\text{pgcd}(a, p) = 1$, $a^{p-1} = 1 \bmod p$. En multipliant par a on obtient $a^p = a \bmod p$. Il reste le cas où $\text{pgcd}(a, p) > 1$; alors, p divise a, donc $a^p = 0 \bmod p$ et $a = 0 \bmod p$ donc l'égalité $a^p = a \bmod p$ est bien vérifiée. □

Chapitre 6

RSA

6.1 La cryptographie asymétrique

La cryptographie asymétrique, encore appelée cryptographie à clé publique, est une découverte récente dans l'histoire du chiffrement, qui a véritablement révolutionné ce domaine. En effet, d'une part elle a permit de résoudre le problème crucial de l'échange des clés de chiffrement symétrique, et d'autre part elle a apporté une solution au problème de l'authentification.

Les principes de la cryptographie à clé publique ont été établis par Whitfield Diffie et Martin Hellmann, dans un article devenu célèbre intitulé *New directions in cryptography* et publié en 1976. Un an plus tard, un tel système est réalisé en pratique, par Ron Rivest, Adi Shamir et Leonard Adleman, c'est le système cryptographique RSA, toujours utilisé de nos jours. Depuis, plusieurs autres systèmes de cryptographie asymétrique ont été découverts, que nous n'étudierons pas dans le cadre de ce cours.

Décrivons d'abord les principes généraux d'un chiffrement asymétrique. Dans un tel système, les protagonistes possèdent chacun une clé qui lui est propre. Ainsi, dans le schéma classique de communication entre Alice et Bob, Alice possède une clé k_A et Bob possède une clé k_B. Chaque clé est constituée de deux parties : une partie secrète (ou privée) et une partie publique. Ainsi,

$$k_A = (\mathrm{k}_\mathrm{A}^\mathrm{priv}, \mathrm{k}_\mathrm{A}^\mathrm{pub}) \qquad k_B = (\mathrm{k}_\mathrm{B}^\mathrm{priv}, \mathrm{k}_\mathrm{B}^\mathrm{pub}).$$

Comme leur nom l'indique, les clés privées $\mathrm{k}_\mathrm{A}^\mathrm{priv}$ et $\mathrm{k}_\mathrm{B}^\mathrm{priv}$ ne doivent être connues que de leur propriétaire, soit respectivement Alice et Bob, tandis que les clés publiques sont accessibles à tous.

On a de plus deux fonctions : une de chiffrement E et une de déchiffrement D. À la différence d'un système symétrique, elles sont paramétrées chacune par seulement une partie de la clé. Plus précisément, E est paramétrée par la partie publique k^pub et D est paramétrée par la partie privée k^priv. Autrement dit, on demande que *l'application inverse de* $E_{\mathrm{k}^\mathrm{pub}}$ *soit* $D_{\mathrm{k}^\mathrm{priv}}$.

Ainsi, si Alice souhaite envoyer un message à Bob, elle chiffre son message m par :

$$c = E_{\mathrm{k}_\mathrm{B}^\mathrm{pub}}(m).$$

Bob reçoit le chiffré c, et le déchiffre avec sa clé privée :

$$m = D_{\mathrm{k}_\mathrm{B}^\mathrm{priv}}(c).$$

En effet, puisque $D_{\mathrm{k}_\mathrm{B}^{\mathrm{priv}}} = E^{-1}_{\mathrm{k}_\mathrm{B}^{\mathrm{pub}}}$, on a bien

$$D_{\mathrm{k}_\mathrm{B}^{\mathrm{priv}}}(c) = D_{\mathrm{k}_\mathrm{B}^{\mathrm{priv}}}(E_{\mathrm{k}_\mathrm{B}^{\mathrm{pub}}}(m)) = m.$$

Notons que Alice, pour chiffrer m, a utilisé la clé publique de Bob, et n'a pas eu besoin de sa clé privée.

En retour, si Bob souhaite répondre à Alice, il utilisera la clé publique d'Alice pour chiffrer sa réponse, et Alice devra déchiffrer à l'aide de sa propre clé privée.

Pour que ce système soit sûr, l'opposant Oscar ne doit pas pouvoir déchiffrer les messages chiffrés qu'il peut intercepter. Pour cela, *il est nécessaire qu'il ne puisse pas calculer la partie privée d'une clé à partir de sa partie publique* (à laquelle il a accès bien sûr). Ainsi, il ne doit pas pouvoir calculer $\mathrm{k}_\mathrm{B}^{\mathrm{priv}}$ à partir de $\mathrm{k}_\mathrm{B}^{\mathrm{pub}}$ ou $\mathrm{k}_\mathrm{A}^{\mathrm{priv}}$ à partir de $\mathrm{k}_\mathrm{A}^{\mathrm{pub}}$.

En général, la sécurité d'un système cryptographique asymétrique repose sur *la difficulté calculatoire d'une opération mathématique*. Par exemple, RSA repose sur la difficulté de la factorisation des grands entiers.

6.2 Le chiffrement RSA

Définition 6.2.1. Une clé RSA $k = (\mathrm{k}^{\mathrm{pub}}, \mathrm{k}^{\mathrm{priv}})$ est définie à partir des paramètres suivants :
- p et q sont deux (grands) nombres premiers distincts et de même taille binaire.
- e et d sont des entiers tels que $ed = 1 \bmod (p-1)(q-1)$.
- $N = pq$.

Alors

$$\mathrm{k}^{\mathrm{pub}} = (N, e) \quad \text{et} \quad \mathrm{k}^{\mathrm{priv}} = d.$$

Les messages clairs et chiffrés sont identifiés à des éléments de $\mathbb{Z}/N\mathbb{Z}$. Les fonctions de chiffrement et de déchiffrement sont définies par :

$$\begin{array}{rcl} E_{\mathrm{k}^{\mathrm{pub}}} : \mathbb{Z}/N\mathbb{Z} & \to & \mathbb{Z}/N\mathbb{Z} \\ m & \mapsto & m^e \bmod N \end{array} \qquad \begin{array}{rcl} D_{\mathrm{k}^{\mathrm{priv}}} : \mathbb{Z}/N\mathbb{Z} & \to & \mathbb{Z}/N\mathbb{Z} \\ c & \mapsto & c^d \bmod N \end{array}$$

Théorème 6.2.2. Avec les notations précédentes, on a bien, pour tout $m \in \mathbb{Z}/N\mathbb{Z}$,

$$D_{\mathrm{k}^{\mathrm{priv}}}(E_{\mathrm{k}^{\mathrm{pub}}}(m)) = m.$$

Démonstration. On va supposer pour simplifier la démonstration, que m est un entier premier à N, autrement dit que $m \in (\mathbb{Z}/N\mathbb{Z})^*$. D'après les hypothèses, $ed = 1 \bmod (p-1)(q-1)$, donc il existe un entier a tel que $ed = 1 + a(p-1)(q-1)$. On a :

$$\begin{aligned} D_{\mathrm{k}^{\mathrm{priv}}}(E_{\mathrm{k}^{\mathrm{pub}}}(m)) = D_{\mathrm{k}^{\mathrm{priv}}}(m^e) &= \left(m^e\right)^d \bmod N \\ &= m^{ed} \bmod N \\ &= m^{1+a(p-1)(q-1)} \bmod N \\ &= m\left(m^{(p-1)(q-1)}\right)^a \bmod N. \end{aligned}$$

Puisque $N = pq$, et que p et q sont des premiers distincts, d'après la proposition 5.2.3, $\varphi(N) = \varphi(pq) = (p-1)(q-1)$. D'après le théorème d'Euler 5.3.1, comme on suppose $\mathrm{pgcd}(m, N) = 1$,

$$m^{(p-1)(q-1)} = 1 \bmod N.$$

Donc

$$D_{\mathrm{k}^{\mathrm{priv}}}(E_{\mathrm{k}^{\mathrm{pub}}}(m)) = m\big(m^{(p-1)(q-1)}\big)^a = m \bmod N.$$

□

Remarque 6.2.3. Plaçons nous maintenant du point de vue de l'opposant Oscar. Celui-ci connait e et N, mais pas d, qui lui servirait à déchiffrer. Supposons maintenant que Oscar soit capable de factoriser N. Cela signifie qu'il peut calculer les nombres premiers p et q tels que $N = pq$. Alors il peut en déduire $c := (p-1)(q-1)$, puis, en effectuant l'algorithme d'Euclide étendu entre e et c, il peut calculer $d = e^{-1} \bmod c$. Il est donc nécessaire, pour que le chiffrement RSA soit sûr, que N ne puisse pas être factorisé. Dans la pratique, on choisit les paramètres de sorte que l'entier N soit de l'ordre de 1024 bits. Le système RSA repose sur la dissymétrie calculatoire des deux opérations suivantes, en quelque sorte inverses l'une de l'autre :

1. Choisir deux nombres premiers p et q de 512 bits et calculer $N = pq$
2. Étant donné N un entier de 1024 bits ayant exactement deux facteurs premiers, calculer ces facteurs.

Il existe des algorithmes rapides pour effectuer la première opération, alors que la deuxième est impossible à réaliser (en un temps raisonnable) avec les méthodes connues à l'heure actuelle.

Remarque 6.2.4. Le chiffrement RSA, ou un autre système de chiffrement à clé publique, ne peut pas se substituer complètement à un système de chiffrement symétrique, car les opérations de chiffrement et de déchiffrement sont trop lentes pour être utilisées sur des données un tant soit peu importantes. Dans la pratique, on utilise un système hybride, dans lequel les deux types de chiffrement coexistent, le chiffrement asymétrique étant utilisé une seule fois pour échanger la clé d'un algorithme de chiffrement symétrique, lequelle est utilisé ensuite pour la durée d'une session de communications. Ainsi, le chiffrement asymétrique est utilisé pour résoudre le problème de l'échange des clés symétriques.

6.3 L'authentification RSA

Supposons maintenant que Bob veuille transmettre un message m à Alice, en garantissant à Alice qu'il est bien l'auteur de ce message. Il s'agit ici d'une problématique, l'authentification, totalement différente de celle de la confidentialité. En effet, cette fois, Bob ne souhaite pas empêcher l'opposant Oscar de prendre connaissance de m. Il souhaite empêcher Oscar *de pouvoir se faire passer pour lui*, par exemple en communicant avec Alice en prétendant être Bob.

Voici comment il peut procéder : il calcule $s := D_{\mathrm{k}_{\mathrm{B}}^{\mathrm{priv}}}(m)$ et transmet à Alice le couple (m, s). Alice, pour se convaincre que le couple (m, s) provient bien de Bob, vérifie que

$$E_{\mathrm{k}_{\mathrm{B}}^{\mathrm{pub}}}(s) = m.$$

En effet, on a

$$E_{\mathrm{k}_{\mathrm{B}}^{\mathrm{pub}}}(s) = m \iff s = D_{\mathrm{k}_{\mathrm{B}}^{\mathrm{priv}}}(m)$$

ce qui montre à Alice que s a bien été obtenu à l'aide de $\mathrm{k}_{\mathrm{B}}^{\mathrm{priv}}$, que Bob est la seule personne à détenir.

Chapitre 7

Codes correcteurs : une introduction

7.1 Introduction

Avec ce chapitre, nous quittons la cryptographie pour aborder le problème de *la correction d'erreurs*. Nous allons considérer la situation suivante : un message binaire $x \in \{0,1\}^k$ transite sur un canal de communications qui est suceptible de modifier x. Notre but est de protéger x de l'action du canal. On peut penser à un disque compact, dont les poussières ou les rayures vont perturber l'information binaire qu'il contient, ou encore à une communication téléphonique soumise à des interférences. Plus précisément, nous allons faire les hypothèses suivantes, sur le type de perturbations que le canal peut occasioner : on note y le message obtenu à l'issue de la communication.

1. Le message x peut subir un ou plusieurs *effacements* : un de ses bits est perdu, mais la position des bits perdus est connue. Par exemple, $x = 0110$ et $y = *110$. Ici, x a subi un effacement sur son premier bit.
2. Le message x peut subir une ou plusieurs *erreurs* : certains de ses bits sont changés, mais on ne sait pas lesquels (sauf si on connait x bien sûr). Par exemple, $x = 0110$ devient $y = 1111$. Ici, deux erreurs ont affecté x.

Afin de minorer l'effet du canal, l'idée de base est de rajouter de la *redondance* au message x, avant de le transmettre au travers du canal de communications, de telle sorte que x puisse être retrouvé à partir du message reçu, tout au moins lorsque la perturbation causée par le canal n'est pas trop importante. Nous commençons par quelques exemples :

7.1.1 Le bit de contrôle de parité :

Soit $x \in \{0,1\}^k$. On rajoute à x un *bit de contrôle de parité* $x_{k+1} \in \{0,1\}$ de sorte que le nombre de 1 dans $x' = (x_1, \ldots, x_{k+1})$ soit pair. Par exemple :

$$x = 1010 \qquad x' = 1010\mathbf{0}$$
$$x = 1110 \qquad x' = 1110\mathbf{1}$$

On peut alors corriger un effacement dans x'. En effet, la valeur du bit effacé doit restaurer la parité du nombre de 1. Par exemple,

si $y' = 111 * 00$ c'est que $x' = 111100$ et donc $x = 11110$.

Par contre, si x' subit une erreur, la parité du nombre de 1 n'est plus respectée dans le message reçu y', toutefois on ne peut pas savoir à la lecture de y' quel bit a été affecté. Par exemple,

si $y' = 100$ c'est que $x' \in \{000, 110, 101\}$.

On voit que le bit de contrôle de parité *corrige un effacement, détecte un erreur mais ne corrige pas une erreur.*

Afin de pouvoir corriger une erreur, l'idée est d'introduire plusieurs bits de parité associés à des parties différentes du message. Pour illustrer cette idée on va voir deux exemples.

7.1.2 Le code carré

Ici $x \in \{0,1\}^4$ et on introduit un bit de parité associé aux indices $\{1,2\}$, $\{3,4\}$, $\{1,3\}$, $\{2,4\}$. Il est plus adéquat d'écrire x sous la forme d'un tableau :

$$\begin{array}{cc} x_1 & x_2 \\ x_3 & x_4 \end{array}$$

alors on ajoute un bit de parité pour chaque ligne et chaque colonne de ce tableau. Illustration avec $x = 1101$. Les bits de parité sont retranscrits en caractère gras.

$$\begin{array}{ccc} 1 & 1 & \mathbf{0} \\ 0 & 1 & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \end{array}$$

On dit que $x = 1101$ est *encodé* en $x' = 1101\mathbf{0110}$.

Si x' subit deux effacements on peut alors les corriger. En effet, on voit que au moins un effacement est seul sur une ligne ou une colonne ; on peut alors utiliser cet ensemble de parité pour le corriger. Illustration : supposons que nous ayons reçu

$$\begin{array}{ccc} * & * & \mathbf{0} \\ 0 & 0 & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \end{array}$$

La première colonne nous indique que $x_1 = 1$. Pour retrouver $x_2 = 1$ on utilise soit la deuxième colonne soit la première ligne. Par contre, certaines configurations de trois effacements ne peuvent pas être corrigées, par exemple :

$$\begin{array}{ccc} * & 1 & * \\ 0 & 0 & \mathbf{0} \\ * & \mathbf{1} & \end{array}$$

que l'on peut corriger de deux façons :

$$\begin{array}{ccc} 0 & 1 & \mathbf{1} \\ 0 & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \end{array} \quad \text{ou} \quad \begin{array}{ccc} 1 & 1 & \mathbf{0} \\ 0 & 0 & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \end{array}$$

On peut se convaincre que ce système d'encodage permet de corriger une erreur en raisonnant sur les parités qui ne sont pas satisfaites. En effet, si l'erreur est sur l'un des bits de x, alors deux parités seront fausses, une horizontale et une verticale, et l'erreur se trouve à l'intersection. Si l'erreur est sur l'un des bits de parité, une seule parité sera fausse. Par exemple, si on reçoit

$$\begin{array}{ccc} 1 & 0 & \mathbf{0} \\ 0 & 1 & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \end{array}$$

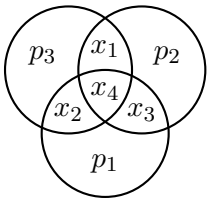
on constate que la première ligne et la deuxième colonne contiennent une erreur donc on sait que l'erreur (si elle est unique) est sur x_2.

Ainsi le codes carré *corrige deux effacements et une erreur*. Avec ce système d'encodage on doit transmettre 8 bits pour 4 bits d'information. On dit que son *rendement* est de $4/8 = 1/2$.

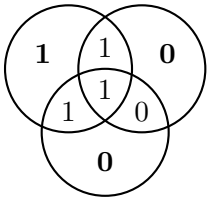
7.1.3 Le code de Hamming

Nous allons maintenant voir un système d'encodage plus économique, puisqu'il permet la correction d'une erreur et de deux effacements avec seulement trois bits de parité pour quatre bits effectifs. Son rendement est donc de $4/7$. Ce système de correction a été inventé par Richard Hamming en 1950, un mathématicien qui travaillait à l'époque sur les ordinateurs du Laboratoire Bell.

On encode un message $x = (x_1, x_2, x_3, x_4)$ en lui rajoutant les bits de parité p_1, p_2, p_3 correspondants aux ensembles de parité indiqués par un cercle dans la figure suivante :



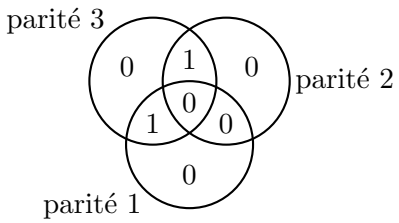
Par exemple, pour $x = 1101$ on obtient :



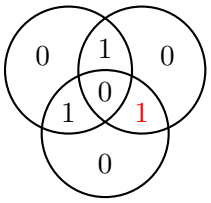
Montrons que l'on peut corriger une erreur. Pour cela, on remarque que, si un bit donné est erroné, les cercles qui l'entourent vont contenir un nombre impair de 1 tandis que ceux auxquels il n'appartient pas ne seront pas affectés, donc contiendront un nombre pair de 1. Il suffit donc de caractériser chaque position par l'ensemble des cercles qui l'entourent. Pour chaque cercle, numéroté par l'indice du bit de parité correspondant, on note 1 s'il contient la position et 0 sinon, et on fait un tableau :

	x_1	x_2	x_3	x_4	p_1	p_2	p_3
cercle 1	0	1	1	1	1	0	0
cercle 2	1	0	1	1	0	1	0
cercle 3	1	1	0	1	0	0	1

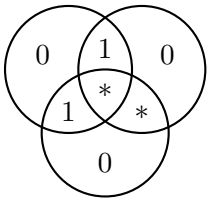
On voit que les colonnes sont deux à deux distinctes. Exemple de correction : supposons que $x' = (x_1, x_2, x_3, x_4, p_1, p_2, p_3)$ soit devenu $y' = 1100000$, et que seulement une erreur ait eu lieu :



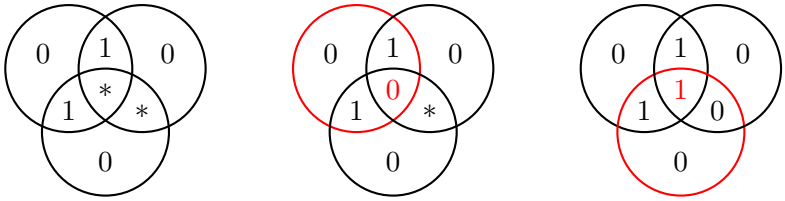
On voit que les parités 1 et 2 ne sont pas respectées mais que 3 est ok. Cela correspond donc à la colonne $\begin{smallmatrix}1\\1\\0\end{smallmatrix}$ du tableau, qui elle-même indique le bit x_3. Il faut donc corriger la valeur de y'_3 pour restaurer les parités et retrouver $x' = 1110000$:



Montrons maintenant que l'on peut corriger deux effacements. En effet, on remarque que deux positions quelconques sont telles qu'il y a toujours un cercle qui contient l'une mais pas l'autre. Cela découle du fait que les colonnes du tableau sont deux à deux distinctes. Par exemple, si on avait eu dans le message x' précédent deux effacements :



on peut utiliser le cercle 3 pour déduire $x_4 = 0$ puis le cercle 1 pour x_3. On obtient x' par les étapes successives :



Le code de Hamming *corrige une erreur et deux effacements avec un rendement de 4/7.*

7.2 Un peu de formalisme matriciel

Nous allons interpréter en termes matriciels ce que nous avons fait au paragraphe précédent, dans le but de motiver les développements du prochain chapitre.

Tout d'abord le bit de parité x_{k+1} associé à $x = (x_1, \ldots, x_k)$ s'interprète agréablement par l'équation suivante :

$$x_1 + \cdots + x_k + x_{k+1} = 0 \bmod 2.$$

Tous les calculs seront désormais effectués modulo 2, conséquemment on omettra la notation mod2 et on écrira simplement :

$$x_1 + \cdots + x_k + x_{k+1} = 0.$$

Notons que cela est équivalent à

$$x_{k+1} = x_1 + \cdots + x_k.$$

Considérons maintenant le codage de Hamming. Nous avons défini une application :

$$\begin{aligned} \{0,1\}^4 &\to \{0,1\}^7 \\ x &\mapsto x' \end{aligned}$$

où, si $x = (x_1, x_2, x_3, x_4)$, $x' = (x_1, x_2, x_3, x_4, p_1, p_2, p_3)$ avec :

$$\begin{aligned} p_1 &= x_2 + x_3 + x_4 \\ p_2 &= x_1 + x_3 + x_4 \\ p_3 &= x_1 + x_2 + x_4. \end{aligned}$$

On peut interpréter x' comme le produit de x par une matrice :

$$x' = (x_1, x_2, x_3, x_4) \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{7.1}$$

Lors de la phase de correction du message reçu $y' = (y_1, y_2, y_3, y_4, y_5, y_6, y_7)$, nous avons calculé les trois parités associées aux trois cercles du schéma puis nous avons cherché dans le tableau la colonne correspondante, c'est-à-dire contenant un 0 pour une parité correcte et un 1 pour une parité incorrecte. Par exemple, celle du cercle 1 est correcte si $y_2 + y_3 + y_4 + y_5 = 0$; donc la première coordonnée de la colonne que nous cherchons vaut exactement $y_2 + y_3 + y_4 + y_5$. En raisonnement de même pour les autres cercles, on voit que la colonne associée à y est :

$$\begin{aligned} &y_2 + y_3 + y_4 + y_5 \\ &y_1 + y_3 + y_4 + y_6 \\ &y_1 + y_2 + y_4 + y_7 \end{aligned}$$

que nous pouvons à nouveau interpréter par un produit de matrices :

$$\begin{pmatrix} y_2 + y_3 + y_4 + y_5 \\ y_1 + y_3 + y_4 + y_6 \\ y_1 + y_2 + y_4 + y_7 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} \tag{7.2}$$

Ainsi, on peut remplacer les raisonnement “géométriques” du paragraphe précédent par des calculs matriciels : l'encodage d'un message x est le résultat du produit de x par une matrice comme dans (7.1) ; pour corriger y' (en supposant qu'une seule erreur a affecté x') on effectue le produit (7.2) et on cherche dans le tableau la colonne correspondante. Celui-ci nous indique quelle coordonnée de y' doit être corrigée. Les avantages de cette approche linéaire sont nombreux. D'une part, il est facile d'imaginer d'autres méthodes d'encodage simplement en changeant de matrice dans (7.1). D'autre part, nous avons à disposition les outils de l'algèbre linéaire et du calcul matriciel pour analyser leurs propriétés. Nous allons voir que les notions de système linéaire, de dimension, de bases, seront très utiles, à condition d'accepter, et c'est un point important, que les résultats et les méthodes d'algèbre linéaire vues sur les corps $\mathbb{R}$ et $\mathbb{C}$ (UE Algèbre 1) s'étendent au *corps à deux éléments* $\{0, 1\}$ muni de l'addition et de la multiplication modulo 2.

Chapitre 8

Espace de Hamming binaire et codes linéaires

8.1 L'espace de Hamming

Définition 8.1.1. On appelle *espace de Hamming binaire* et on note H_n l'ensemble $H_n = \{0,1\}^n$. Un élément de H_n est appelé *un mot* et on dit que n est *sa longueur*.

L'espace de Hamming H_n est un espace vectoriel de dimension n sur le corps fini à deux éléments $\{0,1\}$.

Notation. On note $+$ l'opération d'addition dans H_n : c'est l'addition "bit à bit" notée $\oplus$ dans les chapitres précédents. Par exemple :

$$00111110 + 11110000 = 11001110.$$

On note $\epsilon_1, \ldots, \epsilon_n$ la base canonique de H_n :

$$\begin{aligned} \epsilon_1 &= 1000\ldots0 \\ \epsilon_2 &= 0100\ldots0 \\ &\ldots \\ \epsilon_n &= 0000\ldots1 \end{aligned}$$

Pour $x \in \{0,1\}^n$, on note $\overline{x}$ le mot obtenu à partir de x en inversant tous ses bits. Par exemple si $x = 01000$ alors $\overline{x} = 10111$. On a $\overline{x} = x + 11\ldots1 = x + 1^n$.

On remarque aussi qu'introduire une erreur dans $x \in H_n$ en position i revient à ajouter ϵ_i à x.

Remarque 8.1.2. On admettra que toutes les notions vues en Algèbre 1 dans le cadre des espaces vectoriels sur $\mathbb{R}$ et $\mathbb{C}$ s'appliquent sur $\{0,1\}$ (notions de familles libres, génératrices, de base, de sous-espace vectoriel, de rang, d'application linéaire, etc..).

Définition 8.1.3. Le *support* d'un mot $x \in \{0,1\}^n$ est l'ensemble des positions des 1 dans x. On le note $\mathrm{Sup}(x)$. Par exemple, $\mathrm{Sup}(0011) = \{3,4\}$. Il est clair que la connaissance du support d'un mot détermine complètement celui-ci.

Soit $x, y \in H_n$; on note $x \cap y$ le mot binaire dont le support est l'intersection des supports de x et y. Par exemple, $0011 \cap 1110 = 0010$.

Définition 8.1.4. Le *poids de Hamming* $\mathrm{wt}(x)$ d'un élément $x \in H_n$ est :

$$\mathrm{wt}(x) = |\{i : 1 \leq i \leq n,\ x_i = 1\}| = |\operatorname{Sup}(x)|.$$

La *distance de Hamming* $d_H(x, y)$ entre deux éléments $x, y \in H_n$ est :

$$d_H(x, y) = |\{i : 1 \leq i \leq n,\ x_i \neq y_i\}|.$$

Proposition 8.1.5. On a les propriétés suivantes :

1. $\mathrm{wt}(x) = 0 \Longleftrightarrow x = 00\ldots0 = 0^n$.
2. $\mathrm{wt}(x + y) = \mathrm{wt}(x) + \mathrm{wt}(y) - 2\,\mathrm{wt}(x \cap y)$.
3. $d_H(x, y) = \mathrm{wt}(x - y) = \mathrm{wt}(x + y)$.
4. $d_H(x, y)$ est une distance sur H_n, c'est-à-dire elle vérifie les propriétés caractéristiques suivante :
   (a) $d_H(x, y) = 0 \Longleftrightarrow x = y$
   (b) $d_H(x, y) = d_H(y, x)$
   (c) $d_H(x, z) \leq d_H(x, y) + d_H(y, z)$ (inégalité triangulaire).

Démonstration. L'inégalité triangulaire se déduit de l'égalité 2. □

Définition 8.1.6. L'espace de Hamming H_n est muni d'un *produit scalaire* :

$$x \cdot y = x_1y_1 + \cdots + x_ny_n = \sum_{i=1}^{n} x_iy_i \in \{0, 1\}.$$

Si $x \cdot y = 0$, on dit que x et y sont *orthogonaux*.

Proposition 8.1.7. Le produit scalaire $x \cdot y$ vérifie les propriétés suivantes :

1. $x \cdot y = y \cdot x$.
2. $(\lambda x + \mu y) \cdot z = \lambda(x \cdot z) + \mu(y \cdot z)$.
3. $x \cdot (\lambda y + \mu z) = \lambda(x \cdot y) + \mu(x \cdot z)$.
4. On a équivalence de :
   (a) $x = 0^n$
   (b) Pour tout $y \in H_n$, $x \cdot y = 0$.

Démonstration. Les propriétés 1. 2. 3. sont faciles. L'implication $(a) \Rightarrow (b)$ est évidente. Pour démontrer la réciproque, on applique (b) à $y = \epsilon_i$ et on remarque que $x \cdot \epsilon_i = x_i$. □

8.2 Codes linéaires

Définition 8.2.1. Un *code binaire de longueur n* est un sous-ensemble C de $\{0, 1\}^n$. Si C est un sous-espace vectoriel de $\{0, 1\}^n$, on dit que C est un code *linéaire*.

Exemple 8.2.2. $C = \{0^n\}$ et $C = \{0, 1\}^n$ sont deux codes linéaires. $C = \{000, 100, 010\}$ est un code de longueur 3 qui n'est pas linéaire.

Définition 8.2.3.
- La *distance minimale* d'un code C est l'entier noté $d(C)$ et défini par :
$$d(C) = \min\{d_H(x,y) : x \in C, y \in C, x \neq y\}.$$
- Si C est un code linéaire, il a une dimension que l'on note k. Si sa distance minimale est d, on note ses paramètres $[n,k,d]$.

Proposition 8.2.4. Si C est un code linéaire, sa distance minimale vérifie :
$$d(C) = \min\{\mathrm{wt}(x) : x \in C, x \neq 0\}.$$

Démonstration. $d_H(x,y) = \mathrm{wt}(x-y)$ et $z := x - y \in C$. □

Exemple 8.2.5. $C = \{0^n\}$ a pour paramètres $[n,0,0]$. $C = \{0,1\}^n$ a pour paramètres $[n,n,1]$. $C = \{000, 110, 101, 011\}$ est linéaire et a pour paramètres $[3,2,2]$.

Remarque 8.2.6. Un code linéaire de dimension k possède 2^k éléments. En effet, il possède une base à k éléments $e_1, \ldots, e_k$ et ses éléments sont les $\lambda_1 e_1 + \cdots + \lambda_k e_k$ avec $\lambda_i \in \{0,1\}$. Chacune de ces combinaisons linéaires définit un et un seul mot du code donc il y en a 2^k.

Définition 8.2.7. Soit C un code linéaire de dimension k et de longueur n. Une *matrice génératrice* de C est une matrice G à k lignes et n colonnes, dont les lignes forment une base de C.

Remarque 8.2.8. Un code linéaire ne possède pas une seule matrice génératrice. En effet, il en possède autant que de bases.

Proposition 8.2.9. Soit C un code linéaire et soit G une matrice génératrice de C. On a l'équivalence :

1. $x \in C$
2. Il existe $u \in \{0,1\}^k$ tel que $x = uG$.

Démonstration. En effet, si on note $e_1, \ldots, e_k$ les lignes de G, on a $u_1 e_1 + \cdots + u_k e_k = uG$. □

Chapitre 9

Code dual d'un code linéaire

9.1 Définitions

Définition 9.1.1. Soit C un code linéaire de longueur n. On note $C^\perp$ et on appelle *code dual de* C l'ensemble

$$C^\perp = \{x \in \{0,1\}^n : x \cdot y = 0 \text{ pour tout } y \in C\}.$$

Théorème 9.1.2. Si C est un code linéaire de longueur n et de dimension k, alors $C^\perp$ est un code linéaire de longueur n et de dimension $n-k$.

Démonstration. Soit $\{e_1, \ldots, e_k\}$ une base de C. Si $x \in C^\perp$, alors $x \cdot e_i = 0$ pour tout $i = 1, \ldots, k$. Réciproquement, montrons que, si $x \cdot e_i = 0$ pour tout $i = 1, \ldots, k$, alors $x \in C^\perp$. En effet, tout élément $y \in C$ s'écrit $y = u_1 e_1 + \cdots + u_k e_k$, et $x \cdot y = u_1(x \cdot e_1) + \cdots + u_k(x \cdot e_k) = 0$. On a démontré que :

$$C^\perp = \{x \in \{0,1\}^n : x \cdot e_i = 0 \text{ pour tout } i = 1, \ldots k\}.$$

Soit G la matrice dont les lignes sont les e_i. La matrice G est donc une matrice génératrice de C. On remarque que

$$Gx^t = \begin{pmatrix} e_1 \cdot x \\ e_2 \cdot x \\ \vdots \\ e_k \cdot x \end{pmatrix}$$

où x^t est le vecteur colonne transposé de x. Donc $C^\perp$ est le noyau de l'application linéaire :

$$\begin{aligned} f : \{0,1\}^n &\to \{0,1\}^k \\ x &\mapsto Gx^t \end{aligned}$$

En particulier, cela montre que $C^\perp$ est un sous-espace vectoriel de $\{0,1\}^n$, donc un code linéaire de longueur n. L'image de f est l'espace vectoriel engendré par les colonnes de G donc sa dimension est égale au rang de G qui vaut k. Par le théorème du rang, $\dim(C^\perp) = n-k$. □

Remarque 9.1.3. Attention : contrairement à ce qui se passe sur $\mathbb{R}$ ou $\mathbb{C}$, les espaces C et $C^\perp$ ne sont pas toujours en somme directe. On peut même avoir l'inclusion $C \subset C^\perp$ par exemple pour $C = \{0000, 1111\}$.

Proposition 9.1.4. On a $(C^\perp)^\perp = C$.

Démonstration. Il est clair que $C \subset (C^\perp)^\perp$. On obtient l'égalité en comparant les dimensions de ces deux sous-espaces vectoriels. □

Définition 9.1.5. Soit C un code linéaire de dimension k et de longueur n. Une *matrice de parité* de C est une matrice H à $n-k$ lignes et n colonnes, dont les lignes forment une base de $C^\perp$.

Proposition 9.1.6. Soit C un code linéaire et soit H une matrice de parité de C. On a l'équivalence :

1. $x \in C$
2. $Hx^t = 0^{n-k}$.

Démonstration. Résulte de l'égalité $(C^\perp)^\perp = C$. □

Remarque 9.1.7. Pour décider si un mot $x \in \{0,1\}^n$ appartient à C, il est plus pratique d'utiliser le critère de la proposition 9.1.6 que celui de la proposition 8.2.9.

Corollaire 9.1.8. Soit C un code linéaire de matrice de parité H. Soit $w \in \{1,\dots,n\}$. Alors C contient un mot de poids w si et seulement si H contient w colonnes dont la somme est nulle.

Démonstration. Soit $c_1,\dots,c_n$ les colonnes de H. On remarque que

$$Hx^t = x_1c_1 + \cdots + x_nc_n.$$

Autrement dit, Hx^t est la somme des colonnes de H dont les indices appartiennent au support de x. □

Nous allons voir maintenant une méthode algorithmique qui permet de calculer une matrice de parité d'un code C à partir d'une matrice génératrice. Celle-ci est essentiellement basée sur la méthode du pivot de Gauss vue en Algèbre 1.

9.2 La forme ligne échelonnée d'une matrice.

Dans cette partie, on se place sur un corps de coefficients K quelconque. Dans les applications ultérieures à l'étude des codes binaires linéaires on reviendra au cas $K = \{0,1\}$. On va préciser des résultats vus en Algèbre 1 avec la méthode du pivot de Gauss.

Définition 9.2.1. soit M une matrice à k lignes et n colonnes. Soit, pour $i = 1,\dots,k$, L_i sa ligne d'indice i, et soit j_i la position du premier coefficient non nul de L_i. Si tous les coefficients de L_i sont nuls on pose $j_i = \infty$. On dit que M est *sous forme (ligne) échelonnée* s'il existe un indice s tel que :

1. $L_{s+1} = \cdots = L_k = 0^n$ (i.e. $j_i = \infty$ pour $i = s+1,\dots k$).
2. $1 \leq j_1 < j_2 < \cdots < j_s \leq n$.
3. $M_{i,j_i} = 1$ pour $i \leq s$

Si, de plus, pour $i = 1,\dots,s$, la colonne d'indice j_i contient $M_{i,j_i} = 1$ comme seul coefficient non nul, alors on dit que M est *sous forme (ligne) échelonnée réduite.*

Remarque 9.2.2. Si M est sous forme échelonnée réduite, elle contient une sous-matrice identité de taille s correspondant aux lignes $1, \ldots, s$ et aux colonnes $j_1, \ldots, j_s$.

Définition 9.2.3. On appelle *opérations élémentaires sur les lignes de M* les opérations suivantes :

1. Échange de deux lignes
2. Remplacement de la ligne L_i par λL_i avec $\lambda \neq 0$
3. Remplacement de la ligne L_i par $L_i + \lambda L_j$ avec $j \neq i$.

Proposition 9.2.4.
1. Si M est sous forme échelonnée, alors $s = \text{rang}(M)$.
2. Si la matrice M' est obtenue à partir de M par une succession d'opérations élémentaires alors les lignes de M et les lignes de M' engendrent le même sous-espace vectoriel de K^n.
3. Par une successions d'opérations élémentaires on peut transformer toute matrice M en une unique matrice sous forme échelonnée réduite, que l'on appelle *la forme échelonnée réduite de M*.

Démonstration. On décrit l'algorithme permettant de transformer M en une matrice échelonnée (à suivre sur un exemple) :

On considère la première ligne L_1. Si $L_{1,1} \neq 0$, on pose $j_1 = 1$. Si $L_{1,1} = 0$, on cherche un coefficient non nul en dessous, dans la première colonne. Si on en trouve un, on échange la ligne L_1 avec cette ligne. Si on n'en trouve pas, c'est que la première colonne est nulle et on passe à $L_{1,2}$. On continue ainsi jusqu'à trouver le premier indice de colonne j_1. Si $j_1 = \infty$ c'est que la matrice M est nulle et on a fini. Sinon, on remplace L_1 par $L_1/L_{1,j_1}$ pour que $L_{1,j_1} = 1$. Les colonnes d'indice $1, \ldots, j_1 - 1$ sont nulles. Pour $i = 2, \ldots, k$, on remplace L_i par $L_i - L_{i,j_1} L_1$ afin de mettre des zéros sous L_{1,j_1}.

On a fini avec la première ligne. On itère l'étape précédente sur la sous-matrice d'indice de lignes $2, \ldots, k$ et d'indice de colonnes $j_1 + 1, \ldots, n$, et ainsi de suite jusqu'à la dernière ligne.

La matrice M est maintenant sous forme échelonnée. Pour obtenir sa forme échelonnée réduite, il faut mettre à zéro les coefficients au-dessus des pivots L_{s,j_s} en remplaçant, pour $i = 1, \ldots, s-1$, L_i par $L_i - L_{i,j_s} L_s$. □

9.3 Application aux codes linéaires

9.3.1 Calcul de H à partir de G

Soit G une matrice génératrice d'un code linéaire C de longueur n et de dimension k. Soit G' sa forme échelonnée réduite. Alors G' est aussi une matrice génératrice de C et elle est de rang k. On suppose pour simplifier que $j_1 = 1, \ldots, j_k = k$ c'est-à-dire que G' est de la forme :

$$G' = \begin{pmatrix} I_k & A \end{pmatrix}$$

où A est une matrice à k lignes et $n - k$ colonnes.

Proposition 9.3.1. Sous les hypothèses précédentes, la matrice suivante est une matrice de parité de C :

$$H = \begin{pmatrix} A^t & I_{n-k} \end{pmatrix}$$

Démonstration. Soit C' le code linéaire engendré par la matrice H annoncée. On veut montrer que $C' = C^\perp$. On remarque que $\dim(C') = \text{rang}(H) = n-k$ donc il suffit de montrer l'inclusion $C' \subset C^\perp$, c'est-à-dire que les lignes de H sont orthogonales aux lignes de G'. Soit ℓ_i la ligne d'indice i de G' et h_j la ligne d'indice j de H. Soit $A = (a_{i,j})$. On a :

ℓ_i	0	$\dots$	1	$\dots$	0	$a_{i,1}$	$\dots$	$a_{i,j}$	$\dots$	$a_{i,n-k}$
h_j	$a_{1,j}$	$\dots$	$a_{i,j}$	$\dots$	$a_{k,j}$	0	$\dots$	1	$\dots$	0
indice	1		i		k	$k+1$		$k+j$		n

On voit que $\ell_i \cdot h_j = \sum_{s=1}^{n} \ell_{i,s} h_{j,s} = a_{i,j} + a_{i,j} = 0$. □

Remarque 9.3.2. Si on n'est pas dans le cas où $j_1 = 1, \dots, j_k = k$, on obtient une matrice de parité de C en prenant pour matrice A la matrice extraite de G' correspondant aux colonnes en dehors des j_i, et en positionnant ses lignes aux colonnes $j_1, \dots, j_k$.

9.3.2 Calculer u à partir de x si $x = uG$

Soit C un code linéaire de matrice génératrice G et de matrice de parité H. Si x est un mot de longueur n, on a vu qu'il est facile de décider si $x \in C$ en calculant Hx^t. Si $Hx^t = 0^{n-k}$, on sait qu'il existe $u \in \{0,1\}^k$ tel que $x = uG$. Si la matrice G est sous forme échelonnée réduite les coefficients $u_1, \dots, u_k$ se retrouvent dans x aux positions $j_1, \dots, j_k$. En particulier, si G commence par la matrice identité, ce sont donc les k premiers coefficients de x.

Chapitre 10

Codes linéaires et protection de l'information

Dans ce chapitre, nous allons voir comment on peut utiliser les codes linéaires pour protéger un système de transmission d'information binaire en présence d'effacements ou d'erreurs, en généralisant la méthode introduite au chapitre 7 qui consiste à rajouter de la redondance au message que l'on souhaite transmettre.

10.1 Le codage de canal

On rappelle que l'on souhaite transmettre une information binaire au travers d'un canal qui introduit soit des effacements soit des erreurs. On suppose que le canal agit indépendamment sur chaque bit transmis, en causant un effacement ou une erreur avec une certaine probabilité $p < 1/2$. Dans le premier cas on parle de *canal à effacement* tandis que le deuxième cas est ce qu'on appelle le *canal binaire symétrique*.

On choisit un code linéaire C de paramètres $[n, k, d]$, une matrice génératrice G et une matrice de parité H de C. On rappelle que G est une matrice à k lignes et n colonnes, que H est une matrice à $n - k$ lignes et n colonnes, et que

$$x \in C \iff \text{ il existe } u \in \{0,1\}^k \text{ tel que } x = uG \iff Hx^t = 0^{n-k}. \tag{10.1}$$

On découpe l'information à transmettre en une suite de messages de longueur k. Chacun d'eux est ensuite transmis par la procédure suivante :

1. **Encodage :** le message $u \in \{0,1\}^k$ est transformé en un mot x du code C par la formule $x = uG$. En particulier $x \in \{0,1\}^n$.
2. **Transmission :** le mot de code x est transmis au travers du canal. On note $y \in \{0,1\}^n$ le mot reçu à la sortie du canal.
3. **Décodage :** le mot y est transformé en un mot de code $y^* \in C$ par l'algorithme de décodage suivant :

   Effacements : Choisir y^* parmi les solutions du système linéaire $Hy^t = 0^{n-k}$ dont les inconnues sont les coordonnées de y effacées.

   Erreurs : Choisir y^* parmi les mots de code les plus proches de y pour la distance de Hamming.

Puis, on calcule $u^* \in \{0,1\}^k$ tel que $y^* = u^*G$. Le destinataire du message $u \in \{0,1\}^k$ reçoit $u^* \in \{0,1\}^k$.

Remarque 10.1.1. Dans le cas des erreurs, cette procédure de décodage s'appelle *le décodage par maximum de vraisemblance*. Si les messages u sont équiprobables, elle maximise la probabilité que $y^* = x$, sachant y reçu.

On représente traditionnellement cette procédure appelée *codage de canal* par le schéma suivant :

Définition 10.1.2. Le quotient $R = k/n$ s'appelle *le taux de transmission* du code C. Il mesure le rapport entre les tailles respectives du message d'information effectif et celle du mot transmis, c'est-à-dire le coût de la méthode de transmission.

Théorème 10.1.3. Avec les notations précédentes, l'algorithme de décodage reconstruit correctement le mot de code x transmis à travers le canal, si

1. (Effacements) le nombre d'effacements est inférieur ou égal à $d-1$.
2. (Erreurs) le nombre d'erreurs est inférieur ou égal à $\lfloor (d-1)/2 \rfloor$.

Démonstration. Cas des effacements : notons $t \leq d-1$ le nombre d'effacements. Montrons que le système linéaire (à t inconnues) que résoud le décodeur a une solution unique. cela suffit, puiqu'on sait que x est solution. S'il y a une deuxième solution x', alors x et x' sont égaux en dehors des positions d'effacements. Alors $x - x' \in C$ et $wt(x-x') \leq t$. Mais $t \leq d-1$ et par définition C ne contient aucun mot non nul de poids inférieur à d ; c'est donc que $x = x'$.

Cas des erreurs : On pose $y = x + e$ où e est le vecteur d'erreurs : les coordonnées de e qui sont égales à 1 correspondent aux positions des erreurs. Le nombre d'erreurs t est donc $t = \mathrm{wt}(e)$. Montrons que, si $x' \in C$ est tel que $d(y,x') \leq t$, alors $x' = x$. En effet, par l'inégalité triangulaire :

$$d(x,x') \leq d(x,y) + d(y,x') \leq 2t.$$

Par hypothèse, $2t \leq d-1$. Donc $d(x,x') \leq d-1$. Mais x et x' appartiennent au code C qui est de distance minimale d donc on doit avoir $x = x'$. □

10.2 Probabilité d'erreur et théorème de Shannon

La probabilité d'erreur P_{err} dans la procédure de transmission décrite au paragraphe précédent est le maximum, sur tous les messages $u \in \{0,1\}^k$, de la probabilité que u^* soit différent de u, sachant que u est le message envoyé.

En 1948, Claude Shannon a démontré un théorème très important, qui dit en substance la chose suivante : *on peut rendre la probabilité d'erreur aussi petite que l'on veut, en choisissant un code C convenable, et en plus on peut imposer à C un taux de transmission $R = k/n$ fixé, à condition que R soit inférieur à une certaine quantité appelée la capacité du canal.*

On ne démontrera pas le théorème de Shannon dans le cadre de ce cours, car sa démonstration fait appel à des résultats de probabilité dépassant le niveau de L1.

Toutefois, ce théorème a un défaut majeur : il démontre l'existence de codes adéquats, mais ne donne en aucune manière de méthode de construction de tels codes..

10.3 Le décodage par syndrome

Dans ce paragraphe on se place dans le cas du canal binaire symétrique (celui des erreurs) et on considère le problème du décodage de $y \in \{0,1\}^n$. On cherche donc à calculer le (ou les) mots $y^* \in C$ qui sont les plus proches de y.

Définition 10.3.1. Le syndrome de $y \in \{0,1\}^n$ relativement à la matrice de parité H du code C est $\sigma(y) := Hy^t \in \{0,1\}^{n-k}$.

Remarquons que $\sigma(y) = 0^{n-k}$ si et seulement si $y \in C$ et que dans ce cas bien sûr $y^* = y$. Dans le cas général, posons $y = y^* + e$ avec $y^* \in C$. On a $d_H(y, y^*) = \mathrm{wt}(y - y^*) = \mathrm{wt}(e)$ donc on cherche e de poids le plus petit possible. On a $Hy^t = Hy^{*t} + He^t = He^t$ puisque $y^* \in C$, donc

$$\sigma(y) = He^t.$$

Si $H_1, \ldots, H_n$ sont les colonnes de H, alors

$$He^t = e_1H_1 + \cdots + e_nH_n = \sum_{i:e_i=1} H_i$$

autrement dit He^t est la somme des colonnes de H d'indices appartenant au support de e. Donc le mot e de plus petit poids correspond au plus petit ensemble de colonnes de H dont la somme est $\sigma(y)$. On a donc la procédure de décodage suivante :

1. Calculer $\sigma(y)$.
2. Si $\sigma(y) = 0^{n-k}$, poser $y^* = y$.
3. Sinon, déterminer le (un) plus petit ensemble de colonnes de H dont la somme vaut $\sigma(y)$ (il peut y avoir plusieurs solutions).
4. Poser $e = (e_1, \ldots, e_n)$ avec $e_i = 1$ si et seulement si i est une des colonnes de l'ensemble trouvé en 3.
5. Poser $y^* = y + e$.

Remarque 10.3.2. Si on a une grande quantité d'information à transmettre, on peut procéder à un précalcul pour chaque $\sigma \in \{0,1\}^{n-k}$, d'un mot e_σ associé à un plus petit ensemble de colonnes de H dont la somme vaut σ, puis stocker en mémoire tous les couples (σ, e_σ).

Ensuite, pour décoder un mot y reçu, il suffit de calculer son syndrome $\sigma(y) = Hy^t$, puis de chercher dans la liste le mot $e_{\sigma(y)}$ qui lui est associé. Alors $y^* = y + e_{\sigma(y)}$.

Exemple 10.3.3. On prend pour code C le code de matrice de parité

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

et on note ses colonnes $H_1, \ldots, H_5$. On va construire tous les couples (σ, e_σ). On a huit possibilités pour σ puisque $\sigma \in \{0,1\}^3$. Pour chacune, on exprime σ comme somme de colonnes avec le moins possible de termes ; on obtient ainsi le mot e_σ :

σ	$\begin{matrix}0\\0\\0\end{matrix}$	$\begin{matrix}1\\0\\0\end{matrix}$	$\begin{matrix}0\\1\\0\end{matrix}$	$\begin{matrix}0\\0\\1\end{matrix}$	$\begin{matrix}1\\1\\0\end{matrix}$	$\begin{matrix}1\\0\\1\end{matrix}$	$\begin{matrix}0\\1\\1\end{matrix}$	$\begin{matrix}1\\1\\1\end{matrix}$
	$\emptyset$	H_1	H_2	H_3	H_4	H_5	$H_2 + H_3$	$H_3 + H_4$
e_σ	00000	10000	01000	00100	00010	00001	01100	00110

Noter que pour $\sigma = 111^t$, on a plusieurs possibilités : en effet, $111^t = H_3 + H_4 = H_2 + H_5$.

Supposons maintenant que le décodeur reçoive un mot y, par exemple $y = 11111$. Il calcule $\sigma(y) = Hy^t = \begin{smallmatrix}1\\0\\0\end{smallmatrix}$. Le tableau indique que $e_\sigma = 10000$ donc le décodeur calcule $y^* = y + 10000 = 01111$ et c'est un mot du code C le plus proche de y.

Pour terminer le décodage, il faut savoir avec quelle matrice génératrice le message originel a été encodé. Supposons que ce soit

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

(ici on a calculé une matrice génératrice à partir de H en appliquant la proposition 9.1.6). Alors $y^* = (1,1)G$. Donc $u^* = 11$.

10.4 Codes de Hamming

Dans ce paragraphe, on étudie une famille de codes linéaires, les *codes de Hamming de paramètre r*, de distance minimale 3. Ils corrigent donc une erreur et deux effacements. On a déjà rencontré l'un de ces codes, de longueur 7, au chapitre 7.

Définition 10.4.1. Un code de Hamming de paramètre $r \geq 2$ est un code linéaire dont une matrice de parité est une matrice à r lignes et $2^r - 1$ colonnes et dont les colonnes sont non nulles et deux à deux distinctes.

Remarque 10.4.2. Comme il y a exactement $2^r - 1$ éléments non nuls dans $\{0,1\}^r$, et que H a $2^r - 1$ colonnes non nulles et deux à deux distinctes, l'ensemble des colonnes de H est exactement l'ensemble des éléments non nuls de $\{0,1\}^r$.

Exemple 10.4.3. Pour $r = 2$, la longueur est $n = 2^r - 1 = 3$. On trouve pour matrice de parité, à l'ordre près des colonnes :

$$H = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

et donc pour matrice génératrice en appliquant la proposition 9.1.6 :

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

On voit que $C^\perp$ est le code de parité de longueur 3 et C est le code de répétition.

Exemple 10.4.4. Pour $r = 3$, la longueur est $n = 2^r - 1 = 7$. On trouve :

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

L'encodage avec la matrice G s'écrit :

$$uG = (u_1, u_2, u_3, u_4, u_2 + u_3 + u_4, u_1 + u_3 + u_4, u_1 + u_2 + u_4).$$

On reconnait le système d'encodage de Hamming présenté au chapitre 7, section 7.1.3.

Proposition 10.4.5. Un code de Hamming est un code linéaire de longueur $2^r - 1$, de dimension $2^r - r - 1$, et de distance minimale 3.

Démonstration. Soit H une matrice satisfaisant les conditions de la définition 10.4.1. Puisque H contient tous les mots non nuls de $\{0,1\}^r$ (voir la remarque 10.4.2), elle contient en particulier les colonnes de la matrice identité. Donc H est bien de rang r, c'est donc bien une matrice de parité ; ses lignes forment une base de $C^\perp$ qui est donc de dimension r ; donc C est de dimension $(2^r - 1) - r$.

D'après le corollaire 9.1.8, puisque les colonnes de H sont non nulles et deux à deux distinctes, C ne contient pas de mots de poids 1 ou 2, donc $d(C) \geq 3$. D'autre part, C contient bien des mots de poids 3 puisque par exemple H contient les colonnes $(100\dots)^t$, $(010\dots)^t$ et $(110\dots)^t$ dont la somme est nulle. □

Proposition 10.4.6. Soit $n = 2^r - r - 1$ et C un code de Hamming de longueur n et de matrice de parité H. Soit $y \in \{0,1\}^n$. On trouve un mot y^* de C à partir de y en modifiant au plus un bit de y, suivant la procédure :

1. Calculer $\sigma(y) = Hy^t$.
2. Si $\sigma(y) = 0^r$, poser $y^* = y$.
3. Si $\sigma(y) \neq 0^r$, chercher l'indice i de la colonne de H qui vaut $\sigma(y)$. Alors y^* est obtenu à partir de y en changeant son i-ème bit.

En particulier, si $x \in C$ est transmis et si au plus une erreur affecte x, alors la procédure ci-dessus corrige le mot reçu y en $y^* = x$.

Démonstration. Notons H_i la colonne de H d'indice i et ϵ_i le mot ayant une seule coordonnée non nulle en position i. Alors $H\epsilon_i^t = H_i$. Donc, si $Hy^t = H_i$, c'est que $H(y + \epsilon_i)^t = 0^r$ c'est-à-dire que $y^* := y + \epsilon_i \in C$. □

Remarque 10.4.7. L'algorithme décrit dans la proposition précédente est exactement le décodage par syndrome.

Remarque 10.4.8. La proposition ci-dessus montre que les boules de Hamming de rayon 1 centrées aux mots d'un code de Hamming forment une partition de l'espace de Hamming.

Bibliographie

[1] Simon Singh, *Histoire des codes secrets*, JC Lattès éditeur, 1999.

[2] Claude Shannon, *A mathematical theory of communication*, Bell system technical journal (1948).

[3] Claude Shannon, *Communication theory of secrecy systems*, Bell system technical journal, vol. 28 (1949), 656-715.

[4] Gilles Zémor, *Cours de cryptographie*, Cassini éditeur, 2000.